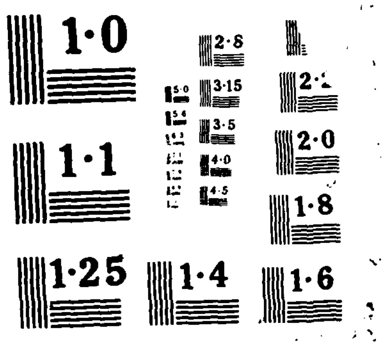
AD-A190 960 PATH FOLLOWING ROBOT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S G GOODWAY DEC 87 1/4

UNCLASSIFIED F/G 12/9 NL


1·0
2·8
2·5
3·15
2·2
5·0
5·6
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

PATH FOLLOWING ROBOT

by

Steven G. Goodway

December 1987

Thesis Advisor George J. Thaler

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b. OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

PATH FOLLOWING ROBOT

12 PERSONAL AUTHOR(S) Goodway, Steven G.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Masters Thesis	FROM ______ TO ______	1987 December	329

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Robot, Robotics, Robot Motion, Path Following

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Given a desired path to be followed by a Robot, a set of commands must be given to the Robot joint servos so that the Robot Tip, or Endpoint, can follow that path. These commands must be synchronized in time and scaled so as to maintain accuracy in the presence of possible saturations in the servos. This Thesis develops an algorithm to generate multiple simultaneous time varying commands to Robot joint positioning servos so that the Robot Tip will follow a desired path for the Cartesian and Articulated Robots. (Keywords: theses; manipulators; computerized simulation; computer programs; models).

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL G. J. Thaler	22b TELEPHONE (Include Area Code) (408) 646-2134; 22c OFFICE SYMBOL 62TR

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

Path Following Robot

by

Steven G. Goodway
Lieutenant, United States Navy
B.S.E.E., University of Washington, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1987

Author: [illegible]
Steven G. Goodway

Approved by: [illegible]
George J. Thaler, Thesis Advisor

[illegible]
Harold A. Titus, Second Reader

[illegible]
John P. Powers, Chairman,
Department of Electrical and Computer Engineering

[illegible]
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

Given a desired path to be followed by a Robot, a set of commands must be given to the Robot joint servos so that the Robot Tip, or Endpoint, can follow that path. These commands must be synchronized in time and scaled so as to maintain accuracy in the presence of possible saturations in the servos. This Thesis develops an algorithm to generate multiple simultaneous time varying commands to Robot joint positioning servos so that the Robot Tip will follow a desired path for the Cartesian and Articulated Robots.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

One of the primary concerns with the use of robotics to do tasks is for that robot to perform the task in real time while maintaining control over the arm with the minimum of interface with the real world. Currently attempts are being made to control robots with elaborate Artificial Intelligence (AI) systems where the robot responds to outside stimuli and uses complex algorithms to perform the next motion. A drawback to this procedure is that it requires a large amount of data to be correlated in a short period of time. In the future as sensors become more accurate and computers speed up so that the data can be handled in a real time fashion then the approach of AI will be effective. Until that time Robots will require pre-designed, pre-determined responses to make the required movement. The accuracy of these motions will depend on the techniques that were used in their construction. Three methods are currently being utilized to teach the Robot to perform the desired task.

A. TEACHING METHODS

The first two methods used in teaching the Robot are Manual Teaching and Lead-Through teaching. Manual teaching is sometimes called teaching-by-showing or guiding and has been used since the early 1950's. Probably the most used method for teaching a robot to move in the desired patterns, it involves manipulating the robot arms, manually, while the joint coordinates are being stored corresponding to each position. Once stored, the program is executed causing the robot to move through the joint vectors in the specified sequence. This method does not require a computer which makes it good for applications that are repetitive in nature and fairly simple such as spot welding, painting, or handling materials.

Lead-through teaching, which is similar to manual teaching, is the simplest for programming continuous path (CP) robot systems. One method is to grab the manipulator and lead it through the entire range of motion at the desired speed, while recording the continuous position of each axis. Due to the size and construction of some robots it may be necessary to have an identical manipulator, equipped with the position sensors, to be led by the operator. Sometimes called a robot simulator or a teaching arm, it is grasped by the operator and led through the range of motion required while the coordinate position of the joints are being sampled at a constant frequency and stored in a computer. Disadvantages to the lead-through teaching method includes 1) requires a simulator; 2) unintentional motions will be recorded and then played back; 3) since it is done manually high precision is not possible; 4) it is impossible to obtain the exact required velocity; and 5) large memory size is required.

B. PROGRAMMING

The third method used to teach robots to perform is through the use of programming. To perform complex maneuvers, or fairly random maneuvers, robots need to be under the control of some form of programming. This is to allow sensors to be positioned in the correct perspective, or data to be retrieved. There are several levels of programming to be considered.

Robot level programming. At this level the computer programming language correlates the sensor data and then specifies the robot motion. This requires that the programmer be familiar with not only the programming language but with techniques used in sensor guided motion. There are several types of robot level programming systems that have been around since the early sixties. They include MHI (1960-1961), WAVE (1970-1975), MINI (1972-1976), AL (1974-present), VAL (1975-PRESENT), AML (1977-PRESENT), TEACH (1975-1978), PAL (1978-PRESENT), MCL (1979-PRESENT), and many others (Lozano-Perez, 1983, pp. 821-841; Fu, Gonzalas, and Lee, 1987, pp. 450-473).

Robot motion by taking the basic concept of guiding and incorporating a decision matrix that is based on sensor input. All of the programming systems require the use of some form of guiding. Basic guiding is the simplest form and takes a sequence of robot positions and repeats them back to the robot. Most of the basic guiding systems are specified by a) the way in which the positions are specified, or b) the method that the robot uses to get from one position to the next. The most common way to specify the position is to a) use a teach-pendant, or b) move the robot through the desired sequence of motions manually or by using some sort of master-slave linkage. When using the teach-pendant method, only a few points are used and the motion is usually straight line between two points relative to a coordinate system. Positions in between are interpolated based on the coordinate system involved. Extended guiding is based on using sensors to tell the position of an object and basing the motion of the robot relative to the coordinate system of that object. Off-line guiding incorporates the task model with the robot model and then simulates the motion of the robot in response to a program or guiding input. This allows for some versatility in choosing the robot path and allows for constraints, such as minimizing time, to be incorporated.

Task level programming. The user defines the task and the robot calculates the necessary parameters to perform the task. This makes the robot entirely independent of the user, requiring no special pre-defined positions or paths that depend on complex geometrical computations to be supplied by the user. Task level programming is accomplished with a task-planner, which, when given a description of the object to be manipulated, initial and final states, and a description of the robot carrying out the task, formulates a robot-level program designed to accomplish the task. Examples of task-level systems include HAND-EYE [Stanford], LAMA [MIT], AUTOPASS [IBM], RAPT, and ROBEX.

C. ROBOT MODEL

So far we have made no mention of the Robot or the type of simulation needed to accurately predict the intended motion of the Robot. Several authors have defined the type and classification (Koren, 1983; Coiffet, 1983, pp.11-37) of robot models that must be considered when selecting a Robot or simulation. Most of the Robots are classified by the type of coordinate system that the arms move in. Of these there are four main classifications:

a. Linear robot model (cartesian), where each of three degrees of freedom move independently, and in a linear fashion. The advantages to this model are the independence of the degrees of freedom reduce the kinematics equation to a simple level, however the model lacks mechanical flexibility. It is a good stepping stone to the more advanced robot models.

b. Cylindrical coordinate robot. Consists of a horizontal arm mounted on a vertical column which is attached to a rotating base. The horizontal arm is capable of moving in and out radially, and up and down the column. The main disadvantage is that the resolution of the movement increases as a function of radial distance.

c. Spherical coordinate robots. Similar to the cylindrical robot however the arm is placed at the end of the vertical column and telescopes in and out. Disadvantage is that the resolution is low in two axes directions. The advantage is that there is more mechanical flexibility and the Robot can access areas below the the base plane.

d. Articulated or Revolute coordinate system where all three of the commanded motions are in terms of angles (radians). This is preferable to the others, but the kinematics, non-linearity of the servos, and time varying torques on each arm, make it a lot more complicated. The main disadvantage is that the joint errors accumulate in the end of the arm. The advantages are that it is the most flexible, and can move at higher speeds than the first three models.

Figure 1.1 (a)-(d) show examples of each of the coordinate systems involved along with skeleton models which interpret the physical motion as a mathematical model. It is easy to see that the specific task a robot is to be used for will be a big factor in determining the type of robot that is needed.

The number of degrees of freedom (d.o.f.) determine the ability of the robot to cover the design space. The most common d.o.f. are two, three, six, and seven. The advantage of two d.o.f. is in the simplicity of motion with the disadvantage of only being able to move in a planar or 2-dimensional coordinate space (2-D). Three d.o.f. give a much more desirable motion in that a full three dimensional space can be considered. Most applications that exist will require as much of 3-dimensional space as possible to be accessible to the tip of the robot. Six degrees of freedom is the desired end product, which consists of the three degree of freedom model with a three degree of freedom manipulator at the end of its arm. This allows for a full range of motion similar to a human hand. A lot of interest is placed in seven d.o.f. as it will not only cover the same motion as six d.o.f. but has the added advantage of more accurately representing the motion of the human body and allows the robot more freedom in moving from one position to the next.

Another important aspect of the Robot model is the kinematic and dynamic equations of motion used to calculate the movement of the robot and describe the dynamic behavior of the manipulator. Several methods are available, each offering a different viewpoint. They are:

a. Kinematics. In the kinematics equations you are given the angles of the arms with respect to the coordinate system and get as a result the position of the end manipulator.

b. Inverse Kinematics. The inverse kinematics equation starts with the position of the end manipulator and results in the calculation of the angles of the arms.

c. Lagrangian. The lagrangian method is based on the calculation of the kinetic energy and the potential energy of the manipulator and is more analytic in nature.

d. Newton Euler Formulation. A numerical method using the center of mass theorem (Newton's) and the kinematic theorem (Euler's) applied to each link of the robot manipulator.

There are several articles and textbooks (Featherstone, 1983, p. 35; Fu, Gonzolas, and Lee, 1987; Paul, 1981, pp. 41-118; Lee, 1981, pp. 62-68; Lee and Ziegler, 1984, pp. 695-696; Lee and Chang, 1986, pp. 1-4; Brady, 1982, pp. 51-126) which go into detail as to which of the equation techniques is the best for the application and which method is easier and quicker to use. Not only do these methods apply to the Robot Model but they also apply to the way that the path of the robot will be calculated.

D. PATH APPLICATION

Movement of the Robot requires that not only do you know where you want to go but that you know where you have been. Starting and ending a series of motions from a pre-defined "home" position allow all of the path motions to originate from a known point with a higher accuracy than if you were to start the motion at the last known point. There is , however a disadvantage in that the arm may be closer to the next desired position and there may be a significant time loss in moving "home" and then back to commence the next series of movements. The end result of the motion must also be considered when determining where a path equation is to go. You need to determine whether or not the paths of the arms and all of the joints need to be programmed or just the tip of the last or furthest arm. That tip is referred to as the endpoint.

When trying to avoid objects it is necessary to know where all of the arms are at all instants. If the object is within the movement space then it may prevent the Robot from reaching some portions of the workspace. Having the Robot tip blindly go to a region where there is no object does not mean that another part of the Robot may not hit the object. If some object is moving in the workspace and must be avoided by the Robot, computation of an acceptable path for the Robot arm may be very difficult. It is problems like these that are driving researchers to AI techniques and away from Robots that do not "think".

E. MOTOR CONTROL

The type of motor control used can be either open loop or closed loop. In an open loop control system the accuracy is based on the ability of the arm sensors to match the desired control input. This is critical since as the load and arm position change the torque required of the motor changes. In the closed loop system, position, velocity, or both are compared with the desired values and the error is fedback to the controlling circuitry so that the error is reduced to minimum. With a highly accurate position servo the type of control becomes less and less a factor. Another factor that affects the accuracy of the system is the type of motor drive. The most accurate are the direct coupled drives while the least accurate indirectly drive the arms through a system of cables, gears, chains, or screws. The type of motor to be controlled must also be considered. Stepping motors, hydraulic actuators, dc servo-motors, or pneumatic cylinders each present unique problems in relationship to the overall system and path design.

Other things to be considered depend on operating specifications. Do you want to handle the problem in minimum time or do you need to conserve energy or cost? Is track accuracy important or is the speed of the task important? unfortunately there is no single answer to all of these problems and again the task determines the design.

F. OBJECTIVES

The objective of this thesis is to have a Robot arm follow a predetermined path. The Path will be defined as a set of coordinate points to be fed into a program which simulates a Robot arm. That is a path that is previously calculated so that all that needs to be done is to feed in the control signals to the Robot arm.

Chapter Two models the Robot Manipulator mathematically without regard to the motor characteristics and considering only the physical characteristics of the arm and the coordinate space that the arm operates in. Due to the simplicity a Cartesian Robot was chosen to perform the movements. Chapter Three takes the motor and derives a computer simulation model to be used in conjunction with the Robot Manipulator of Chapter Two. Chapter Four integrates the chapters into a Dynamic Simulation Language (DSL) program and tests the program in single step movements for proper operation and timing. Chapter Five develops a path following routine which takes cartesian and parametric equations and generates a list of three tuples which are the desired path for the Cartesian Robot to follow. Chapter Six adjusts the earlier Robot simulation, adding control procedures, allowing the path three tuples to be automatically entered into the simulation and demonstrates the Robots ability to follow the generated path. Linear, circular, helical, and sinusoidal motion are examined for proper operation. Also error analysis and criteria for proper operation are discussed.

Chapter Seven takes the same path model for the Cartesian Robot and translates to the "Articulated" or "revolute" coordinate system for the second model to be run. Finally Chapter Eight discusses ideas and directions to be considered for future work.

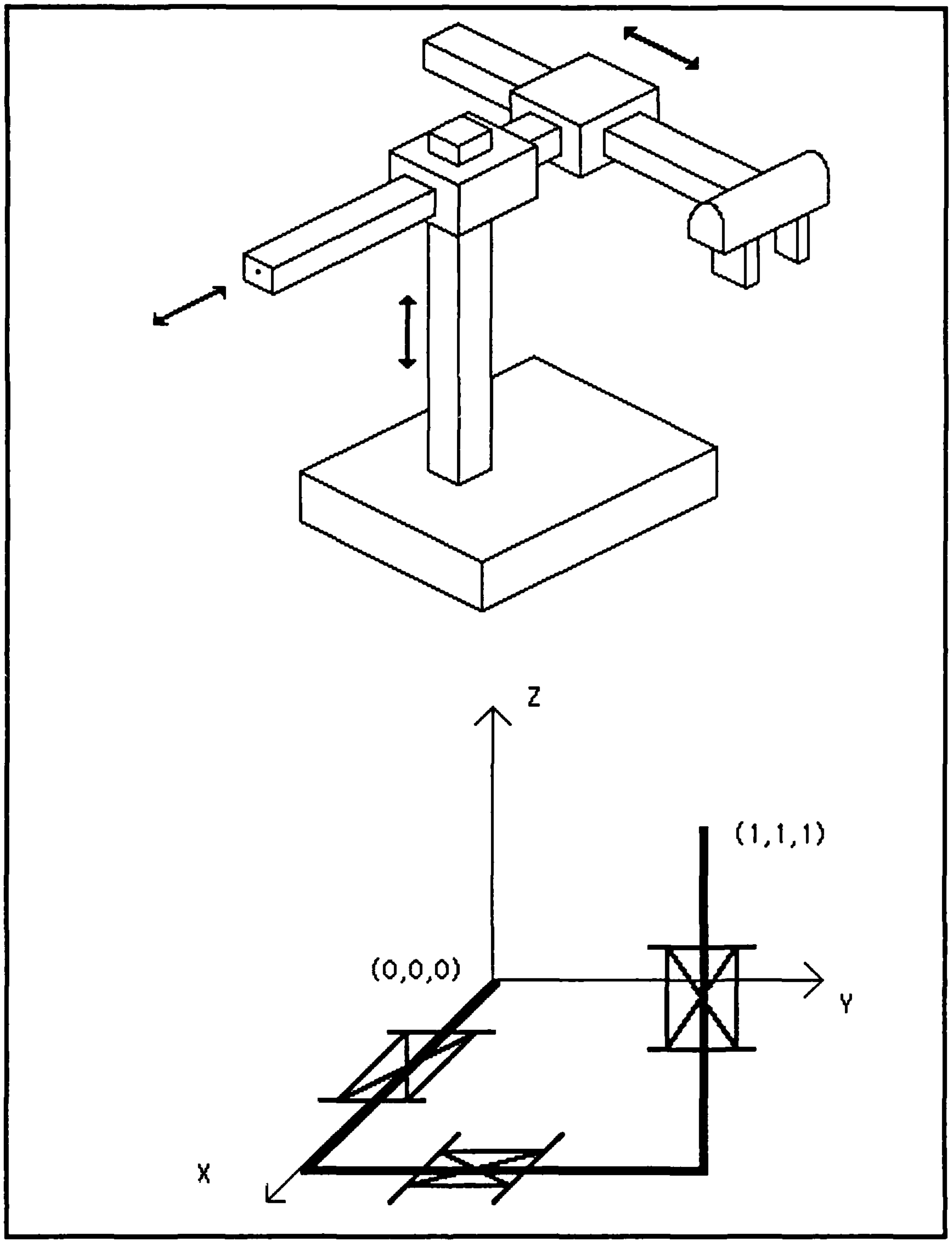


Figure 1.1 Cartesian Robot Model and Skeleton Structure (3-P).

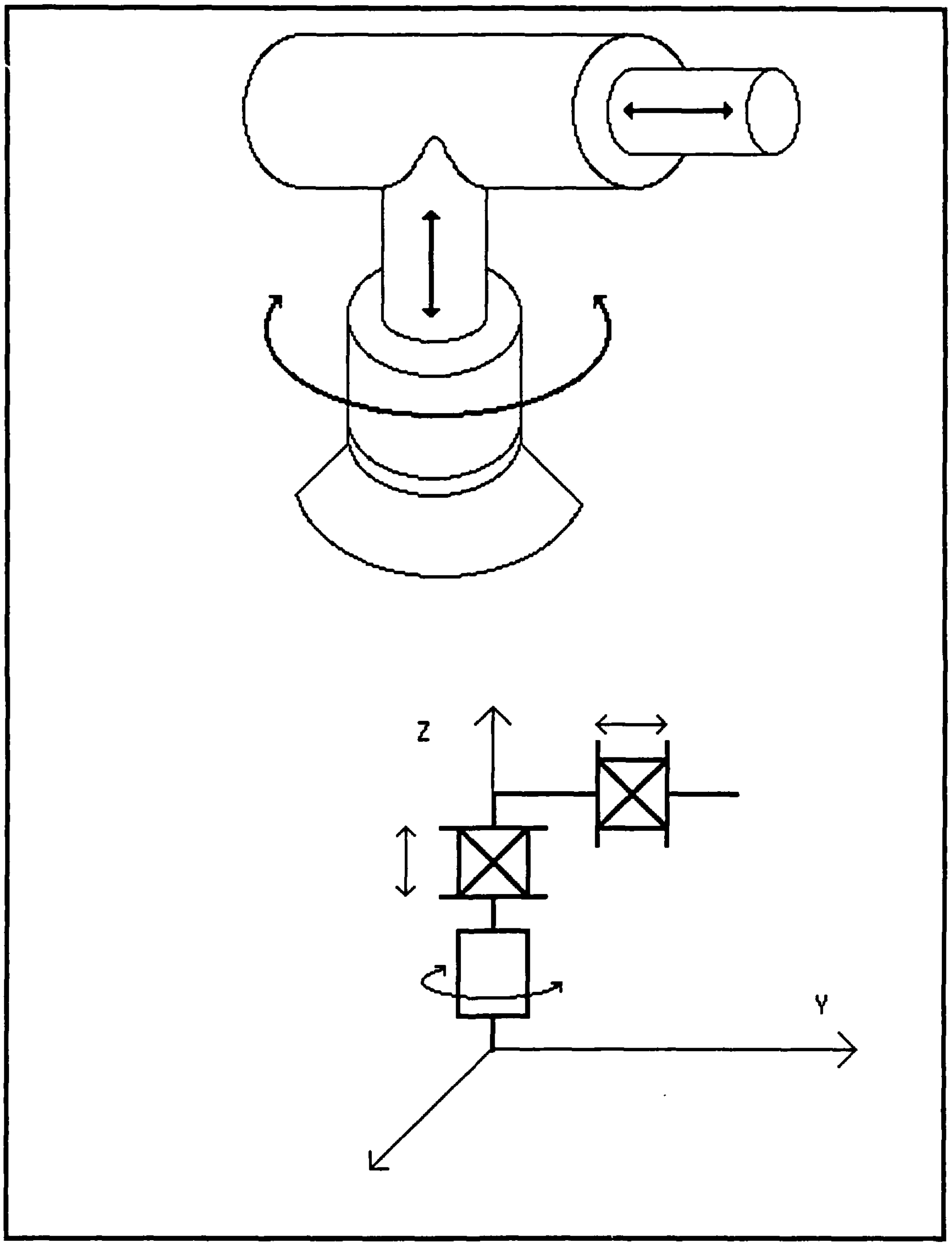


Figure 1.2 Cylindrical Robot Model and Skeleton Structure.

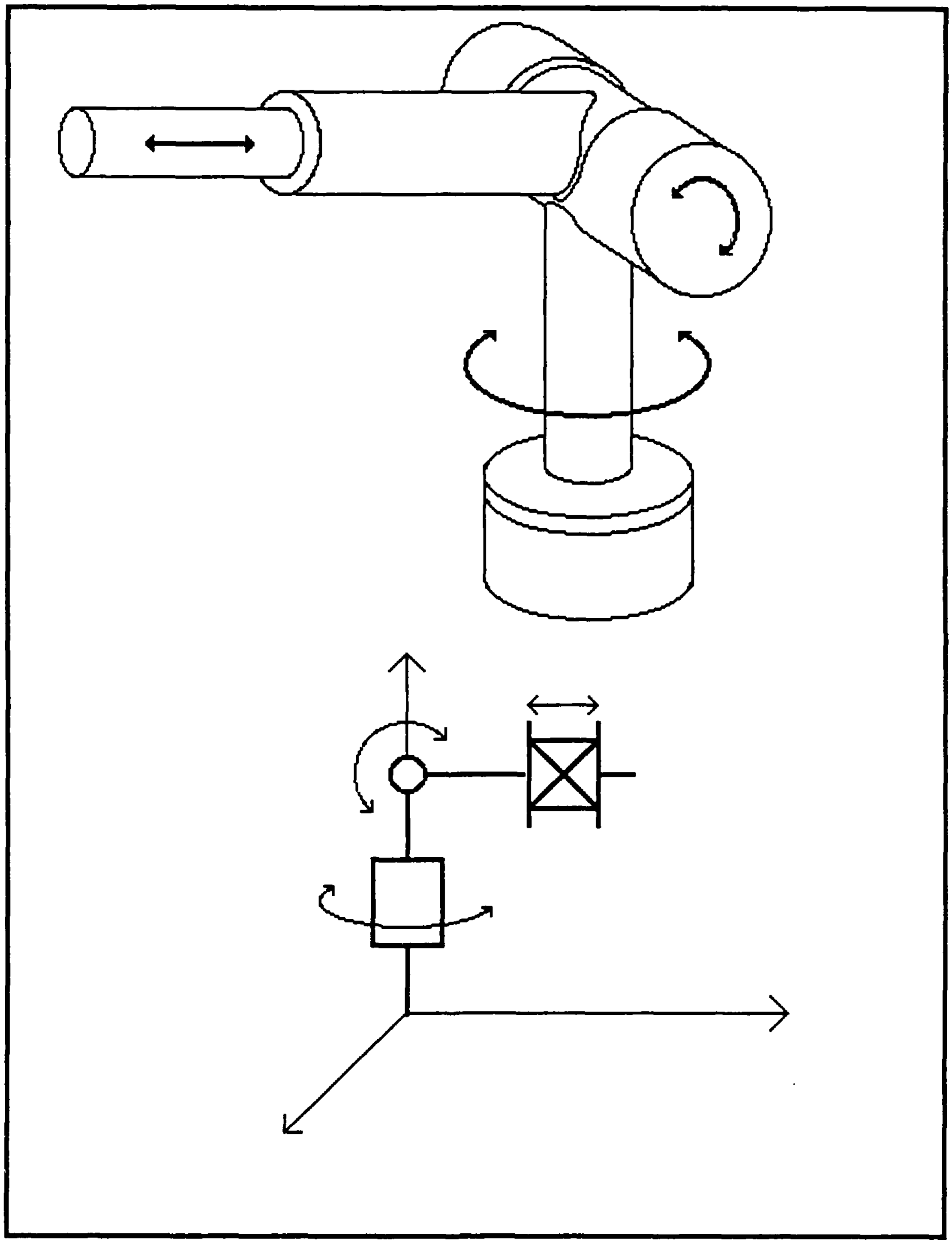


Figure 1.3 Spherical Robot Model and Skeleton Structure.

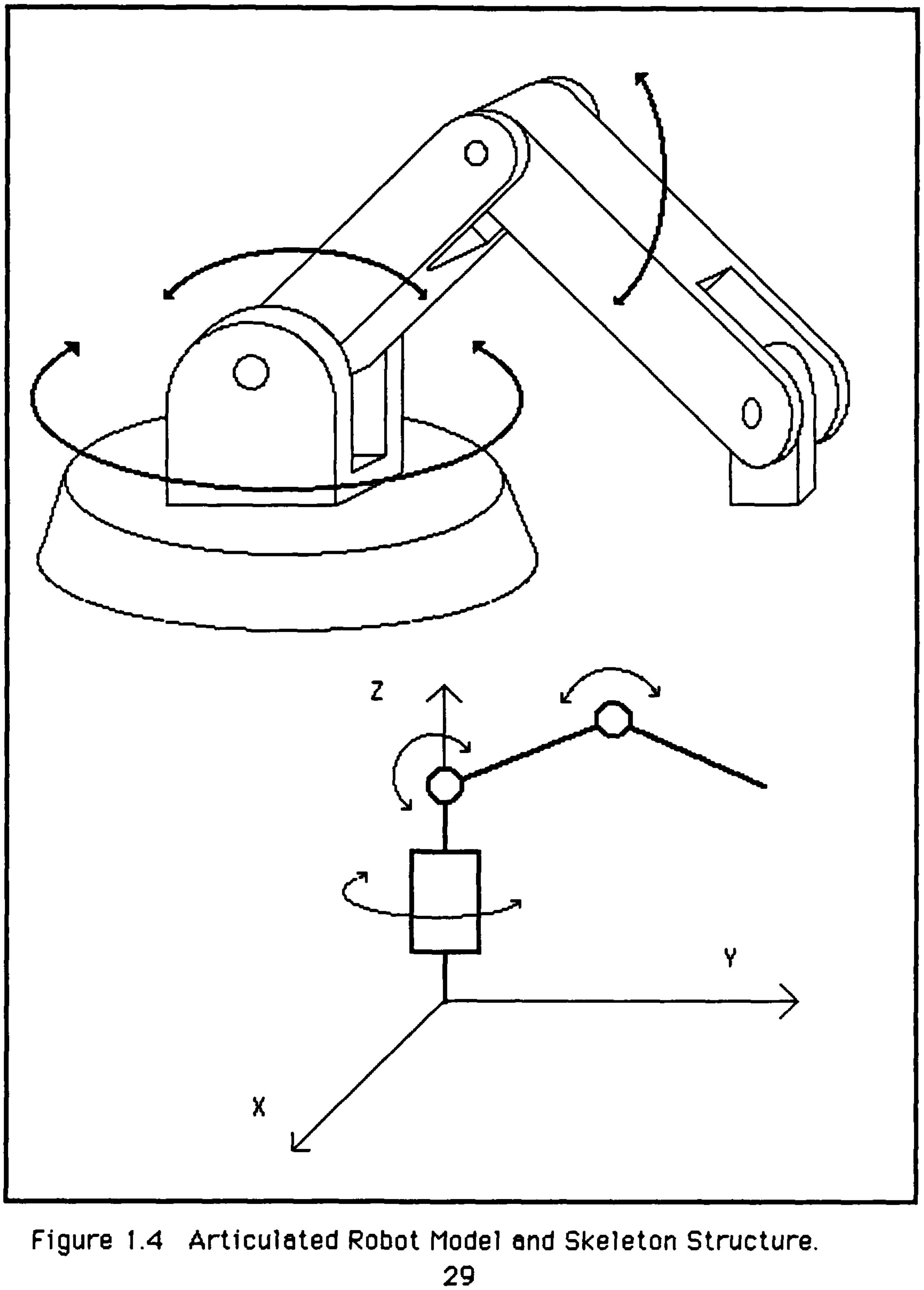


Figure 1.4 Articulated Robot Model and Skeleton Structure.

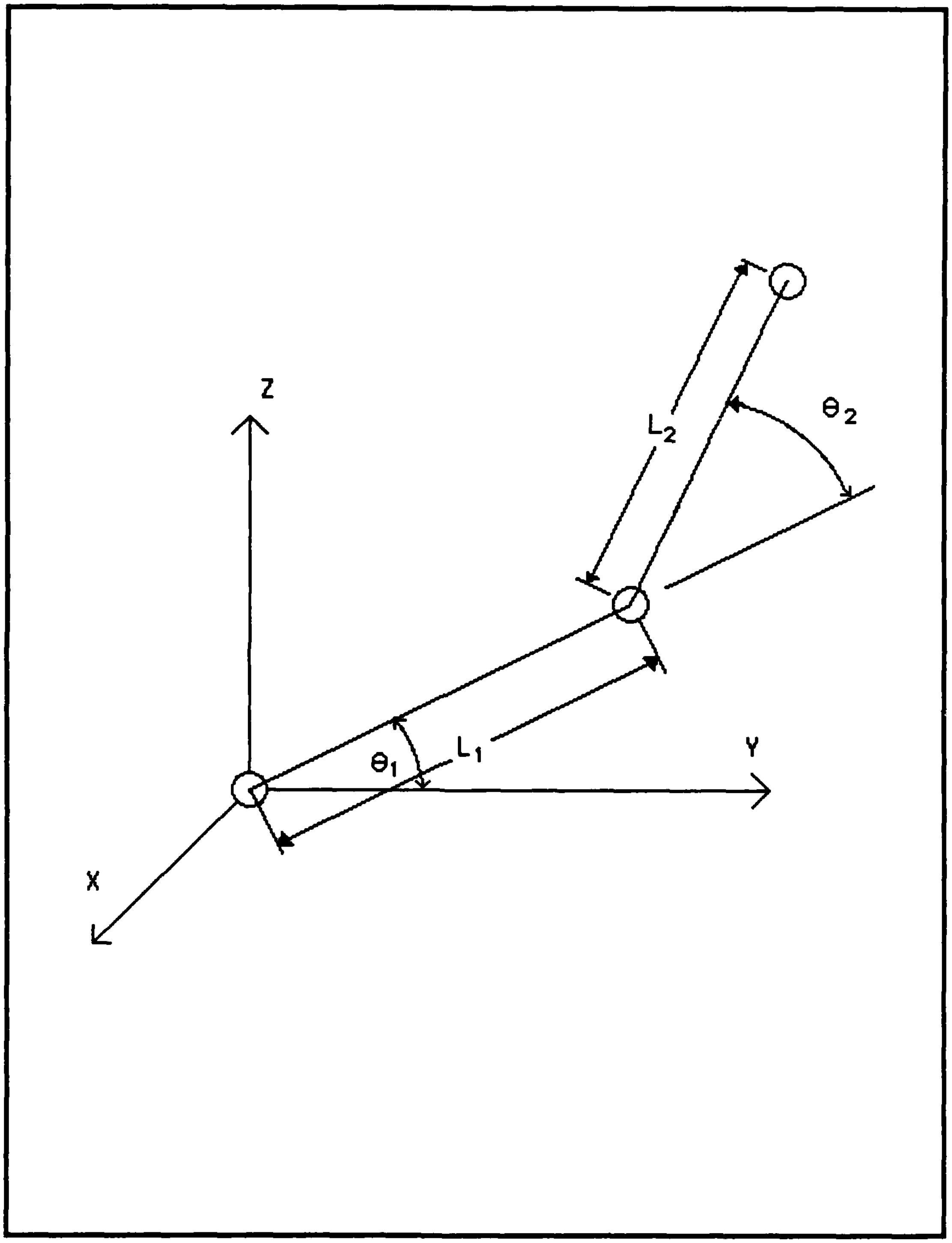


Figure 1.5 Lever Diagram for 2-Dimensional Articulated Robot.

II. MODELING THE ROBOT MANIPULATORS

A. INTRODUCTION

Two Robot manipulators are being considered. The first model consists of a cartesian robot, while the second model consists of a articulated (or revolute) robot. Each mathematical model will be configured with 3 degrees of freedom (d.o.f.) and will be developed using lagrangian equations for both the dynamic and static cases. Parameters of interest include: position, velocity, acceleration, torque, mass, load capabilities, and gravity. The lagrangian relates the the parameters of each joint with all of the other joints. Once the lagrangian has been found for each of the joints, then the equations will be related to the second order model for the idealized servo motor. The cartesian models are being considered first due to the simplicity of the lagrangian equations and because it will be beneficial in understanding the relationships in the articulated robot.

B. CARTESIAN ROBOT MODEL

The 3 d.o.f. cartesian robot that is being considered is shown in Figure 1.1 and whose skeletal structure is displayed in Figure 2.1. The motion of the arms are in the XYZ coordinate space and each of the arms moves in a linear (prismatic) fashion and there are no revolute joints. Because of this they can permit only parallel motion and have no rotation vectors. Using Denavit-Hartenberg Notation (appendix A) (Lee, 1982, pp. 86-70; Fu, Gonzolas, and Lee, 1987, pp. 36-41) on Figure 2.1 we have the following transformation matrices described by link parameters in Table 2.1.

TABLE 2.1
LINK PARAMETERS FOR 3 D.O.F. CARTESIAN ROBOT

Link	Variable	α	a	d	cos α	sin α
1	d_1	0C	0	d_1	1	0
2	d_2	0C	0	d_2	1	0
3	d_3	0C	0	d_3	1	0

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{[eqn 2.1]}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{[eqn 2. 2]}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{[eqn 2. 3]}$$

Computing ${}^0T_3$ from equations 2.1 to 2.3 [appendix A] we get

$${}^0T_3 = \begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad \text{[eqn 2. 4]}$$

which is the description of the end of the manipulator with respect to the base. If the manipulator is referenced to a coordinate system by a transformation C with a tool described by E, the position and orientation of the tool with respect to the coordinate reference is described by X as

$$X = Z\ {}^{0}T_{3}\ E. \qquad \text{[eqn 2.5]}$$

C. DERIVATION OF THE LAGRANGIAN EQUATION OF MOTION

Lagrangian formulation of manipulator dynamics describes the model in terms of work and energy. These dynamic equations can be derived in any coordinate system and are easier to express and use than the Newton-Euler formulation [ASADA]. The lagrangian is defined as the the difference between the kinetic and potential energies of the system

$$L = K - P \qquad \text{[eqn 2.6]}$$

Once the lagrangian has been obtained for each of the joints then the dynamics equation is formulated as

$$Fi = \frac{d}{dt}\frac{\partial L}{\partial \frac{dq_i}{dt}} - \frac{\partial L}{\partial q_i} \qquad i=1\ldots n \qquad \text{[eqn 2.7]}$$

where q_i are in the coordinates in which the kinetic and potential energy are expressed, dq_i/dt is the velocity, and F_i is the torque or force, depending on whether q_i is in a linear or angular coordinate system. The lagrangian of the system is described as L for each degree of freedom n, that is to be modeled.

For the 3 d.o.f. cartesian coordinate system, expressed in cartesian coordinates, all of the joints are linear so that the dynamics equation is expressed in terms of forces, distance d_i, velocity $d\, d_i/d\, t$, and acceleration $d^{\,2} d_i/d\, t^2$. For this problem the lengths of the manipulator arms are identical, with the mass of each arm located at the end of its link.

First Node:

$$K_1 = (1/2)\ m_1\ (d\ d_1/d\ t)^2 \qquad \text{[eqn 2.8]}$$

$$\text{and } P_1 = -m_1 g d_1 = 0 \qquad \text{[eqn 2.9]}$$

To relate the second node to the first it is necessary to first write the terms for the position and then taking the derivative find the velocity components.

Second Node:

$$\begin{array}{ll} x_2 = d_1 & d x_2/d t = d d_1/d t \\ y_2 = d_2 & d y_2/d t = d d_2/d t \\ z_2 = 0 & d z_2/d t = 0 \end{array} \qquad \text{[eqn 2.10]}$$

where

$[v_2]^2 = [dx_2/dt]^2 + [dy_2/dt]^2 + [dz_2/dt]^2$ [eqn 2.11]

$K_2 = (1/2)\, m_1\{(d\, d_1/d\, t)^2 + (d\, d_2/d\, t)^2\}$ [eqn 2.12]

and $P_2 = -m_2 g\, d_2 = 0$ [eqn 2.13]

Third node:

$x_3 = d_1 \qquad dx_3/dt = dd_1/dt$

$y_3 = d_2 \qquad dy_3/dt = dd_2/dt$ [eqn 2.14]

$z_3 = d_3 \qquad dz_3/dt = dd_3/dt$

where

$[v_3]^2 = [dx_3/dt]^2 + [dy_3/dt]^2 + [dz_3/dt]^2$ [eqn 2.15]

$K_3 = (1/2)m_1\{(d\, d_1/d\, t)^2 + (d\, d_2/d\, t)^2 + (d\, d_3/d\, t)^2\}$ [eqn 2.16]

and $P_3 = -m_3 g\, d_3$ [eqn 2.17]

Combining equations [2.8], [2.9], [2.12], [2.13], [2.16], and [2.17] into equation [2.6], and solving we arrive at the lagrangian

$$L = \{1/2\}[(m_1+m_2+m_3)(d\, d_1/d\, t)^2 + (m_2+m_3)(d\, d_2/d\, t)^2 + (m_3)(d\, d_3/d\, t)^2] + m_3 g\, d_3$$ [eqn 2.18]

and calculating the dynamics equation for each node from equation [2.18] we have

First node:

$$\partial L/\partial[d\, d_1/d\, t] = (m_1+m_2+m_3)(d\, d_1/d\, t) \qquad \text{[eqn 2.19]}$$

$$d(\partial L/\partial[d\, d_1/d\, t])/d\, t = (m_1+m_2+m_3)(d^2 d_1/d\, t^2) \qquad \text{[eqn 2.20]}$$

$$\partial L/\partial d_1 = 0 \qquad \text{[eqn 2.21]}$$

we arrive at

$$F_1 = (m_1+m_2+m_3)(d^2 d_1/d\, t^2) \qquad \text{[eqn 2.22]}$$

Second node:

$$\partial L/\partial[d\, d_2/d\, t] = (m_2+m_3)(d\, d_2/d\, t) \qquad \text{[eqn 2.23]}$$

$$d(\partial L/\partial[d\, d_2/d\, t])/d\, t = (m_2+m_3)(d^2 d_2/d\, t^2) \qquad \text{[eqn 2.24]}$$

$$\partial L/\partial d_2 = 0 \qquad \text{[eqn 2.25]}$$

we arrive at

$$F_2 = (m_2+m_3)(d^2 d_2/d\, t^2) \qquad \text{[eqn 2.26]}$$

Third Node:

$$\partial L/\partial[d\, d_3/d\, t] = (m_3)(d\, d_3/d\, t) \qquad \text{[eqn 2.27]}$$

$$d(\partial L/\partial[d\, d_3/d\, t])/d\, t = (m_3)(d\, ^2 d_3/d\, t^2) \qquad \text{[eqn 2.28]}$$

$$\partial L/\partial d_3 = m_3\, g \qquad \text{[eqn 2.29]}$$

we arrive at

$$F_3 = (m_2+m_3)(d\, ^2 d_2/d\, t^2) - m_3\, g \qquad \text{[eqn 2.30]}$$

Equations for F_1 [2.22], F_2 [2.26], and F_3 [2.30], are the lagrangian dynamic equations that we will relate to the servo motor simulation. The values selected for the simulations are

d_1	=	1 unit
d_2	=	1 unit
d_3	=	1 unit
m_1	=	.082 oz + 2mm + load
m_2	=	.041 oz + mm + load
m_3	=	.041 oz
g	=	386.4 in/sec^2
mm	=	0.186
load	=	0.0

A quick review of Figure 2.1 indicates that other variations are available as well. In the calculations we assumed that the arm which allowed X direction of motion actually supports and moves the arms for Y and Z movement. Also the arm which allowed Y motion supported the arm for Z movement. The choice of one unit for each of the arms allows for scaling of the arms to any length configuration. There are many variations but this is the one chosen for the calculations.

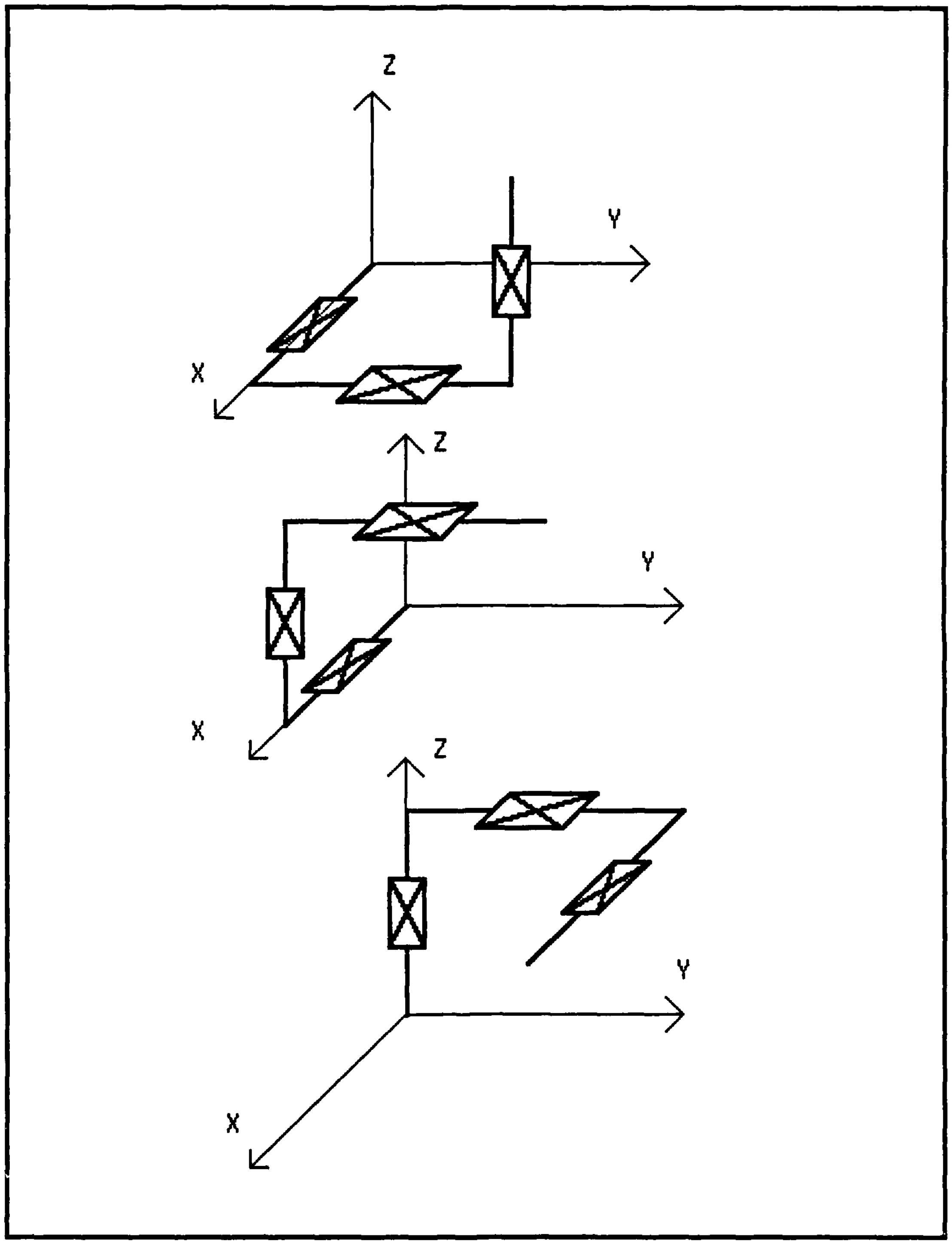


Figure 2.1 Skeleton Orientations for the Cartesian Robot.

III. COMPUTER SIMULATION MODEL

A. INTRODUCTION

The DC servo motor that is being simulated has been demonstrated (Thaler, 1956, pp. 410-417; Ozaslan, 1986, pp.14-15) to have the transfer function

$$\frac{\Theta(s)}{V(s)} = \frac{1/Kv}{S(S(JR/KvKt)+1)(SL/R+1)} \qquad \text{[eqn 3.1]}$$

where			
	$\Theta(s)$	=	Angular position of the shaft
	$V(s)$	=	Applied d-c voltage
	Kv	=	Back emf constant
	Kt	=	Torque constant
	J	=	Total inertia
	R	=	Armature resistance
	L	=	Armature inductance

Since the robot arm has a large inertia when added to the motor inertia this causes the mechanical pole of the motor to become smaller. Coupled with the R/L term, which is large, we can ignore the electrical pole and the transfer function for the motor and robot arm is approximately

$$\frac{\Theta(s)}{V(s)} = \frac{Km}{s^2} \qquad \text{[eqn 3.2]}$$

With this simplification it is easy to see that Figure 3.1 is the block diagram of the second order motor model. As can be seen we are using a velocity feedback loop, an idealized deceleration curve for the motor, and Bang-Bang control to drive the system. Figure 3.2 ties the computer simulation model to that of the idealized motor.

B. COMPUTER MODEL

The computer model uses position feedback while the idealized motor uses velocity feedback of the motor. The input parameter to the computer model is the desired position which drives the ideal motor with the simulated velocity, V. When the E of the computer model has dropped below a minimum value then the arm is in the desired position. Since the deceleration curve of an ideal motor is similar to a parabolic curve it was chosen as the curve to be simulated in the DSL program. The following equations describe the parameters outlined in Figure 3.2 for the computer model.

$$C'' = Km * Vsat \qquad [eqn\ 3.3]$$

$$C' = B\ C''\,dt = Km*Vsat*t + C'(0) \qquad [eqn\ 3.4]$$

$$C = B\ C'\,dt = (Km*Vsat*t^2)/2 + C(0) \qquad [eqn\ 3.5]$$

where $C(0) = 0$. Combining equations 3.4 and 3.5 and solving for C we find the relationship

$$C = (C')^2/(2*Km*Vsat) \qquad [eqn\ 3.6]$$

Other relationships include (for X-arm only):

$E = XPOS - C$ [eqn 3.7]

$X' = - A*K1*(ABS[E])^{1/2}$ [eqn 3.8]

$A = (2*Km*VSAT)$ [eqn 3.9]

where

XPOS is the desired movement in X direction

X' is the commanded velocity

Km is the gain constant :

Km1 = 59.29

Km2 = 90.25

Km3 = 77.44

K1 is the curve scaling constant [0.6]

$X'E = X' - K*C'$ [eqn 3.10]

$V = LIMIT(Vsat,-Vsat,K_2*X'E)$ [eqn 3.11]

where

K is the velocity loop feedback gain [computer model][1.0]

K2 is the amplifier gain [10000.0]

Vsat [150 Volts]

The value of K2 was chosen so that the amplifier saturates for small signals with full values of forward and reverse drive signals [±Vsat].

C. SERVO MOTOR

The second order computer model drives a closed loop-servo for each joint of the robot arm. The motor used to drive each of the robot arms is a permanent magnet motor drive with Table 3.1 listing the parametric characteristics (Ozaslan, 1986, pp. 16-30).

TABLE 3.1
PARAMETRIC DATA FOR JOINT SERVO MOTORS

Torque Constant	Kt	14.4	oz-in/amp
Total Inertia	Jm	0.033	oz-in-sec2/rad
Damping coefficient	Bm	0.04297	oz-in-sec/rad
Back EMF Constant	Kv	0.1012	volts-sec/rad
Armature Inductance	L	100	μ-henries
Avg Terminal Resistance	R	0.91	ohms

The same servo drive unit will be used for all of the joints. Since we have only linear motion at the joints the effective inertia is only the servo motor inertia. For the cartesian robot system there is no reaction torque (no angular acceleration), no centripetal forces (no angular velocity), and no coriolis forces. For each joint we have

$$JTOT1 = J1 + 0.5*(M1 + M2 + M3 + 2*MM + LOAD) \qquad \text{[eqn 3.12]}$$

$$JTOT2 = J2 + 0.5*(M2 + LOAD) \qquad \text{[eqn 3.13]}$$

$$JTOT3 = J3 + 0.5*(M2 + M3 + MM + LOAD) \qquad \text{[eqn 3.14]}$$

where J1, J2, and J3 are the motor drive inertias listed in Table 3.1. The mass values are

M1 = 0.082
M2 = 0.041
M3 = 0.041
MM = 0.186
LOAD = 0.0

Applying the known values to the open loop equation [eqn 3.1] we arrived at the values for KM1, KM2, and KM3.

D. SERVO MOTOR SIMULATION MODEL

Taking into consideration the servo motor and computer model values we can mathematically model the servo motor with the following equations.

$$CR'' = MTE/JTOT \qquad \text{[eqn 3.15]}$$

$$CR' = B\ CR''\,dt = MTE/JTOT *t + C'(0) \qquad \text{[eqn 3.16]}$$

$$CR = B\ CR'\,dt = ((MTE/JTOT) *t^2)/2 + C(0) \qquad \text{[eqn 3.17]}$$

where C(0) = 0. Combining equations 3.16 and 3.17 and solving for C we find the relationship

$$CR = (CR')^2/(2^* \text{MTE/JTOT}) \quad \text{[eqn 3.18]}$$

Other relationships include (for single motor only):

$$Vm = V - Kv^*CR' \quad \text{[eqn 3.19]}$$

$$MP = \text{REALPL}(0.0, L/R, Vm/R) \quad \text{[eqn 3.20]}$$

$$MT = Kt^*MP \quad \text{[eqn 3.21]}$$

$$MTE = MT - Bm^*CR' - TL \quad \text{[eqn 3.22]}$$

where

V is calculated in the computer model

Kv, Bm, Kt are from Table 3.1

TL is the sum of centripetal Torques, Centrifugal Torques, Reaction Torques, and Gravitational Torques which is zero for the cartesian robot arm.

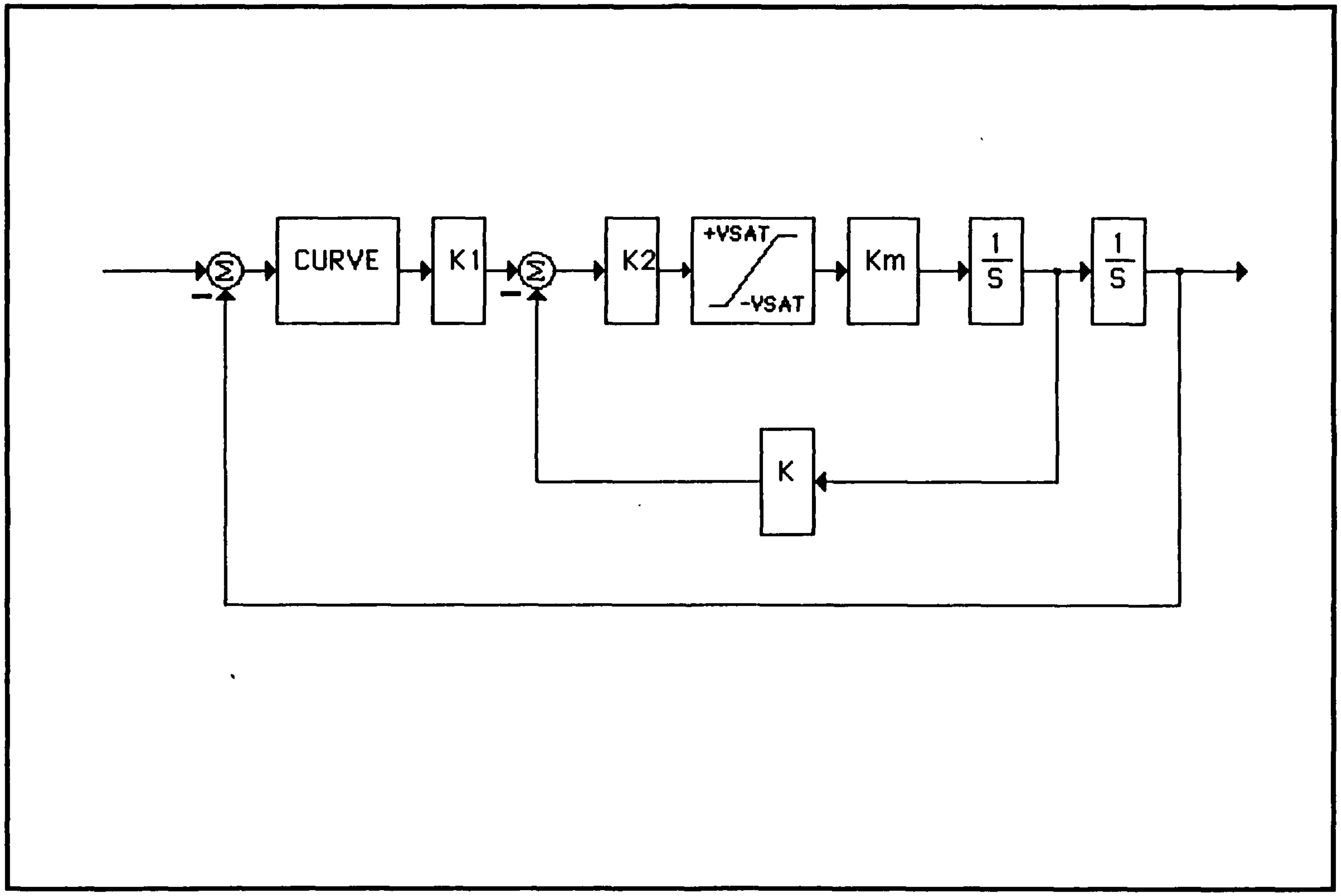


Figure 3.1 Second Order Model with Curve Following Velocity Loop for the Idealized Servo Motor.

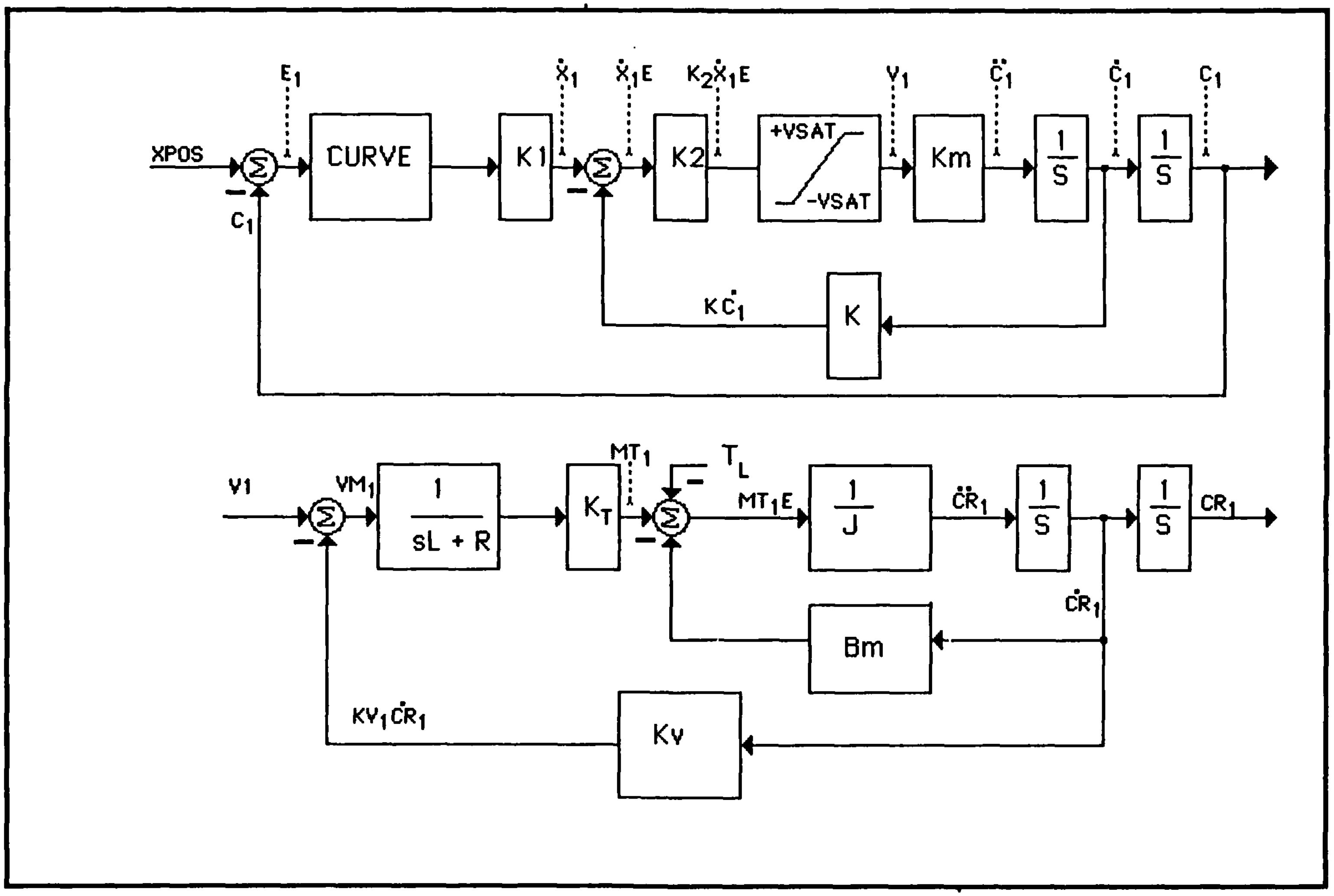


Figure 3.2 Second Order Model with Idealized Servo Motor with Computer Simulation Model.

IV. DSL SIMULATION PROGRAM

A. INTRODUCTION

In Chapter Three the computer model of a robot servo arm was presented, while in Chapter Two a mathematical model was constructed tying the three robot arms together in a cartesian coordinate system. This chapter takes the results of the previous chapters and generates a Dynamic Simulation Language [DSL] Program which will be used to model the entire 3 d.o.f. Cartesian Robot. The input into the DSL program will be a path described by a three tuple of coordinates. The program simulates the robot movement and the next set of coordinate points are inputted. The output of the DSL Program is a listing of the inputted and simulated values based on the method used to input the values.

B. PHILOSOPHY OF OPERATION

The method used to input the data along with the control parameters of the DSL Program determine how the robot will perform and respond to the entered coordinate points. The two methods that are to be considered are the Position Method and the Velocity Method. The Position Method is where the control functions of the DSL Program are controlled by the accuracy of the output position and are independent of time. The Velocity Method fixes the time interval used to input each of the three tuple coordinates and since the input is position oriented we have in effect a velocity control.

The Position Method calculates the robot arm endpoint until the voltages for all of the arms drops to below a stipulated value. This indicates that all of the arms are within the desired values of accuracy. The advantages of the Position Method show up in the accuracy of the final position of each move. Since the step size is predetermined by the programmer and the accuracy of each step is less then the step size then the overall error has a definite maximum limit. The disadvantage with the Position Method is that the values will tend to be very accurate at the expense of time. The values will oscillate around the desired points until all of the voltage values are within a certain minimum value [VALMIN] no matter how long it takes.

The Velocity Method on the other hand follows the same trajectory but when the time interval that was predetermined has been reached the next three tuple of coordinate points are entered. By selecting the maximum velocity available for a given step, the time, determined by the DSL program, and desired length for each step, determined by the computer path program, can be adjusted to maximize the performance at each step. With this adaptability the user can optimize the overall time that it takes to perform the journey or the user can wait at each point. The advantage of this method is in the versatility in controlling with either time, length of travel, or a mixture of both. The disadvantage is in the loss of accuracy at the desired positions but with proper selection of time and length this can be minimized.

C. EXCITATION PROPOSALS

Along with the selection of the method to which the DSL Program is organized there is the type of excitation used to input the desired position. This excitation comes in several forms of which the three of interest are the STEP, RAMP, or a function such as the modified SINE wave. This excitation can also be delayed in time or be a mixture of any of the excitations as desired. Samples of the STEP, Figures 4.1 (a)-(e), and RAMP Excitations, Figures 4.2 (a)-(e) and 4.3 (a)-(e), show differences in the overall response for a single inputted position step of 0.1 units to all three coordinates. The single position change is independent of the time base mentioned earlier but will give an idea of the time required to perform that position change, and the accuracies involved in moving from point to point. Figure 4.4 shows three different excitation techniques used to apply the movement excitation to the DSL Program. First, the STEP excitation is the simplest way to apply the movement to the DSL program. This does not however give a smooth linear response from the simulated robot arm. As seen in Figure 4.2 (c)-(e) the arm movement as simulated comes out fairly non-linear but is still within our earlier assumption that the maximum error for each step is less than the length of the step and looks to be about a tenth of the step size in this case.

The second and third proposed excitations are RAMP excitations and are defined by DSL to be the sum of two RAMPs, the first RAMP starting at time zero (DSL TIME function), and the second RAMP starting at time TV1 but negative in value with the same slope magnitude as the first RAMP. This gives a RAMP from time zero to TV1 with the magnitude at TV1 equal to the same magnitude as the STEP input. Two separate TV1's were chosen to see what the overall effects would be on the simulation arm movement.

Comparing the results of a 0.1 unit position increment, in all three coordinate directions, when excited by a STEP, Figure 4.1(a), and RAMPs, Figure 4.2(a) and 4.3 (a)], show that there is an increase in time for the RAMP to complete its movement and arrive at the desired endpoint. However, while the time increased for the RAMP excitation there is a reduction in the arms velocity as seen in Figures 4.1(b), 4.2(b), and 4.3(b), and the linearity of the arm motion between desired points is much better. RAMP excitations provide much more linear response than STEP excitations and the longer TV1 is then the more accurate a RAMP excitation is. For applications where accuracy is desired it may be more beneficial to use a RAMP excitation with a large TV1. Use of TV1 instead of using a RAMP for the entire step, reduces the complexity of the input structure and attempts to optimize the ramp for both short and long steps. Although most of the steps are nearly the same length there are times when movement in one of the directions is short and requires a faster ramp to complete it in average time. TV1 was chosen as an attempt to keep the speed up while sevicing both long and short steps.

Close examination of the two RAMP excitations show that the longer the RAMP, relative to the time of completion desired, the more linear the travel between points becomes but the longer it takes to get there which is due to the slower velocities that the arms peak at. When designing a system for operation it would be a good idea to incorporate a RAMP technique to lower the velocities of the arms and reduce torques and inertias placed on the motor units along with the added benefit of being more accurate between the inputted steps.

D. MULTIPLE RUNS

The DSL Programming Language can be modified to tie together all of the concepts that have been discussed in this chapter. Some adaptations are necessary to simulate each of the methods. Appendix D contains a copy of all of the modifications used to calculate all of the different DSL simulations that were used in this thesis. Table 4.1 lists all of the modifications that were needed and their location in Appendix D.

TABLE 4.1
DSL MODIFICATIONS AND SIMULATIONS

PROGRAM	METHOD	EXCITATION	LOCATION
ROBSIM1		STEP	APP. D-1
ROBSIM2T	POSITION	STEP	APP. D-3
		RAMP	APP. D-3
ROBSIM2S	VELOCITY	STEP	APP. D-4
		RAMP	APP. D-4

The only difference between the STEP and RAMP excitation programs are in the way the data points are applied to the DSL simulation within the DSL program. Figure 4.4 shows a sample of the STEP and RAMP Excitations along with the equations required to adapt the DSL program for the Ramp Excitation. Figure 4.5 lists the modifications that are needed to change the DSL program from the Position Method of operation to the Velocity Method. Actual test runs will be conducted and referenced in Chapter Six.

E. TEST RUNS

To insure the proper operation of the DSL program it is necessary to make a series of single runs to check on timing and operation for steps of various sizes and negative steps. Table 4.2 lists the DSL programs that were used to test the Robot Model DSL program and shows the step sizes and the time that it took each arm to complete the desired step. Figure 4.6 shows the modifications that were made to ROBSIM1 to make the test runs. For each direction of each step a plot of velocity vs. distance and a plot of distance vs time was done. Due to the number of plots that were done only a representative set of plots are shown.

All of the DSL simulations should give the same time response the averages for each of the step sizes. ROBSIM2 was chosen as the base for the multiple run simulations was due to the control statement and graph ranges.

As can be seen from Table 4.2 there is a distinct relationship between the length of the step size and the time that it takes the robot simulation to reach that value. We will use this relationship when deciding the time that will be used in the Velocity Multiple Run simulations to set up the time interval for repetitive data input and control FINTIME for DSL.

TABLE 4.2

DSL SINGLE RUN MODIFICATIONS

DSL PROGRAM	STEP SIZE (UNITS)	TIME (10^{-3}-SECS) X	Y	Z
ROBSIM1	1.00	40.5	30.0	34.0
	0.50	28.0	20.5	24.0
	0.20	19.0	14.0	15.0
	0.10	10.2	08.5	10.0
	0.05	09.0	06.0	06.5
	0.02	05.0	03.0	04.0
ROBSIM2	0.10	13.0	09.5	11.0
	0.05	09.0	07.0	08.5
	0.02	06.0	04.0	05.0
	0.01	04.0	03.0	03.0
ROBSIM3	0.02	06.0	04.2	05.0
	0.01	04.2	03.0	03.5
	0.005	03.0	02.2	02.5
ROBSIM2N	-0.10	13.0	10.0	10.5
	-0.05	09.0	06.5	07.5
	-0.02	06.0	04.5	05.0

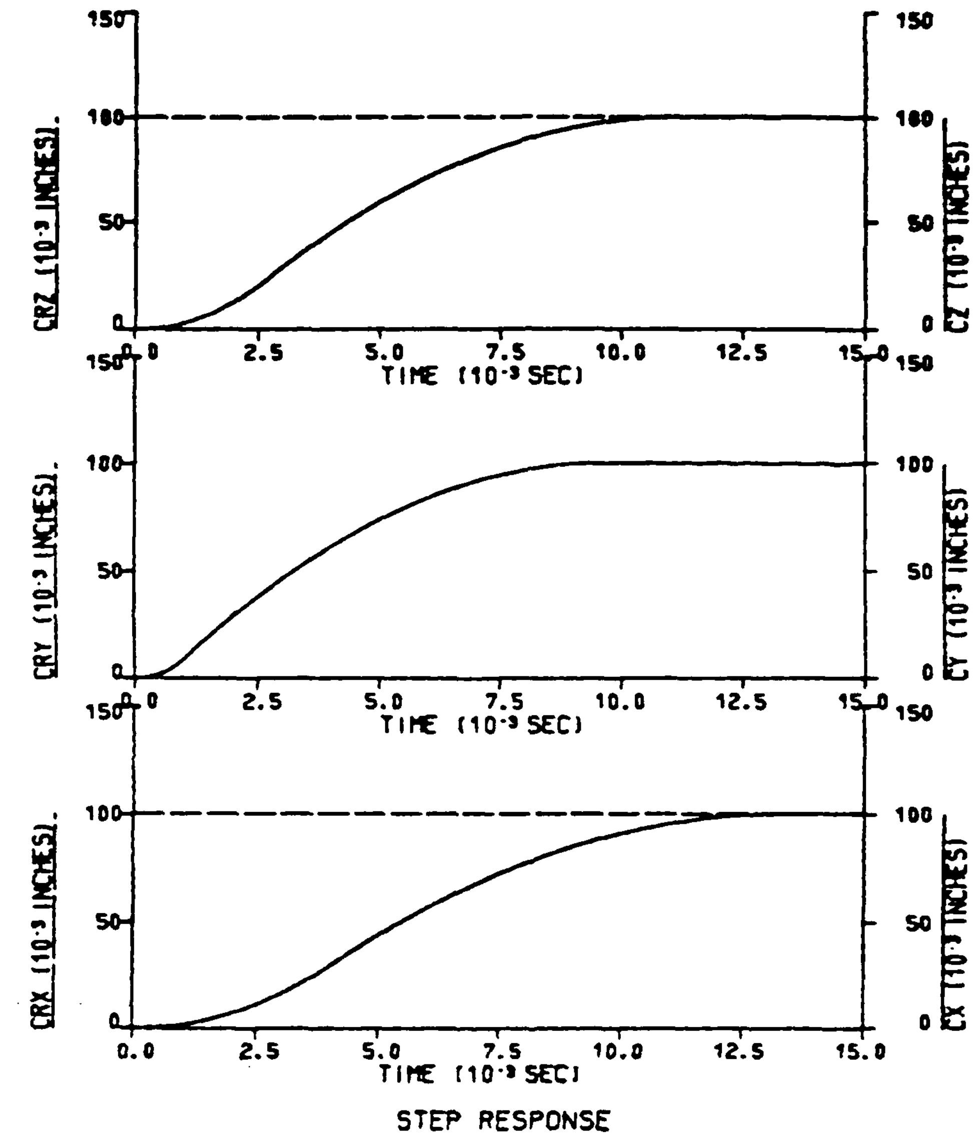


Figure 4.1(a) 0.1 STEP Excited X, Y, and Z Step Response.

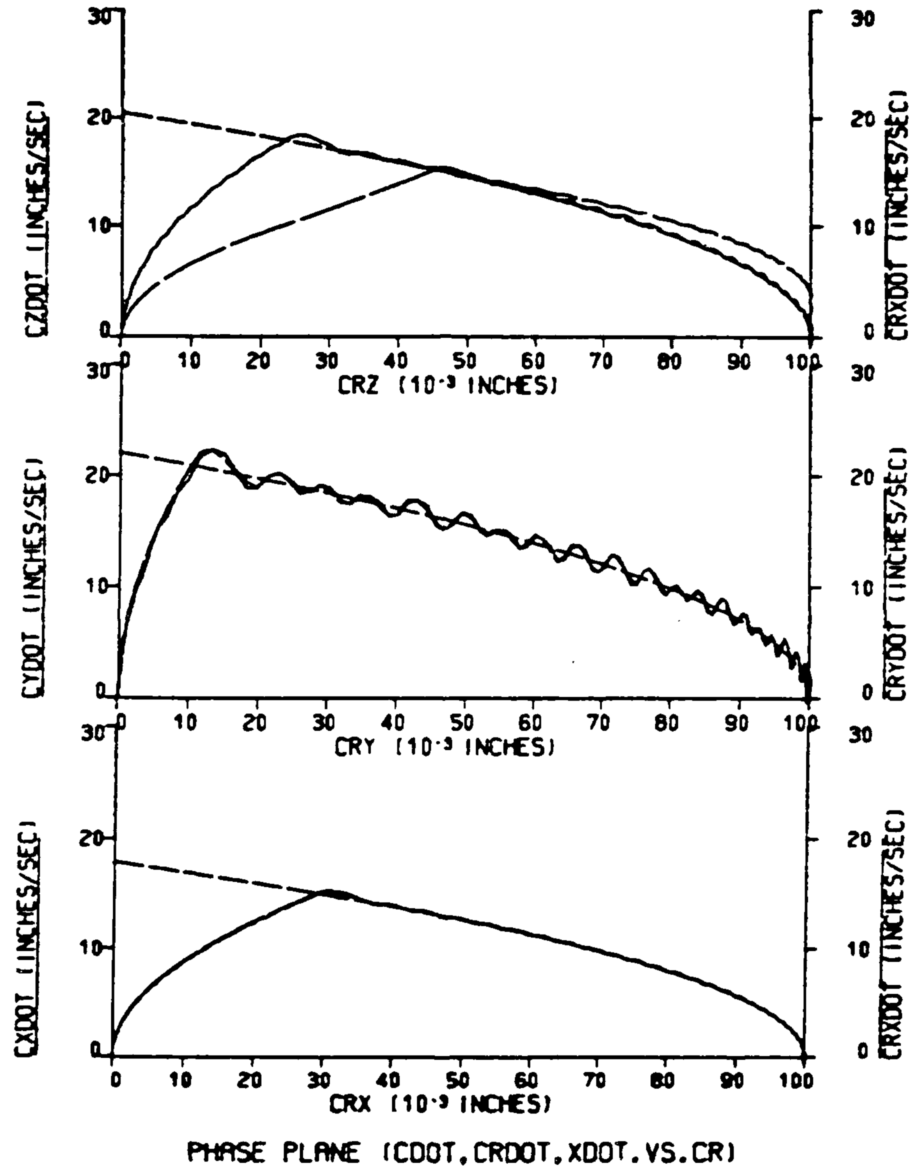


Figure 4.1(b) 0.1 STEP Excited X, Y, and Z Phase Plane.

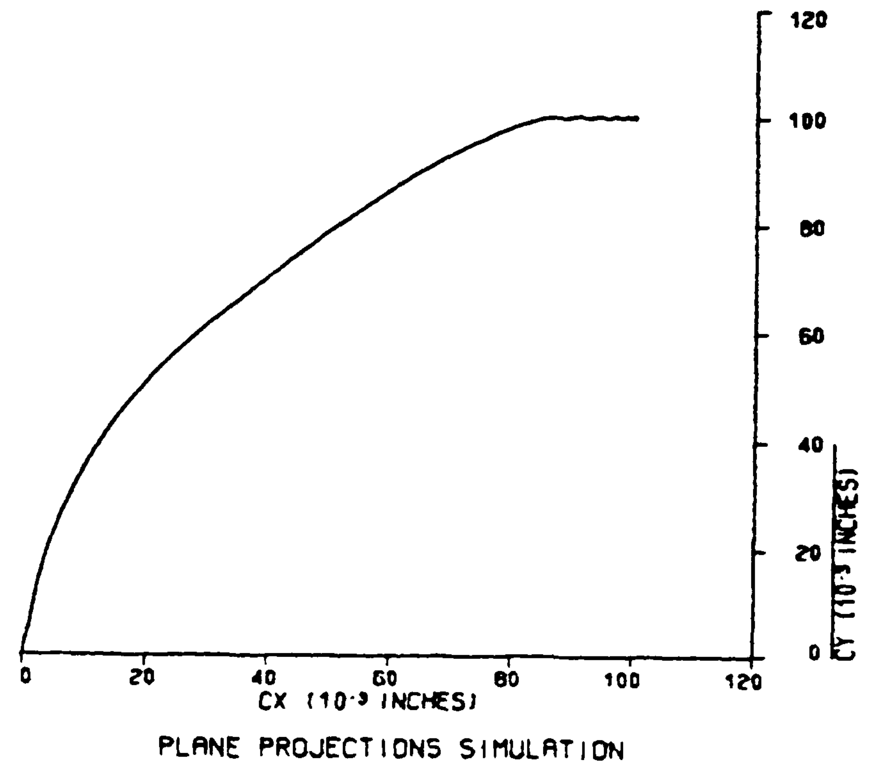


Figure 4.1(c) 0.1 STEP Excited XY Plane Projection.

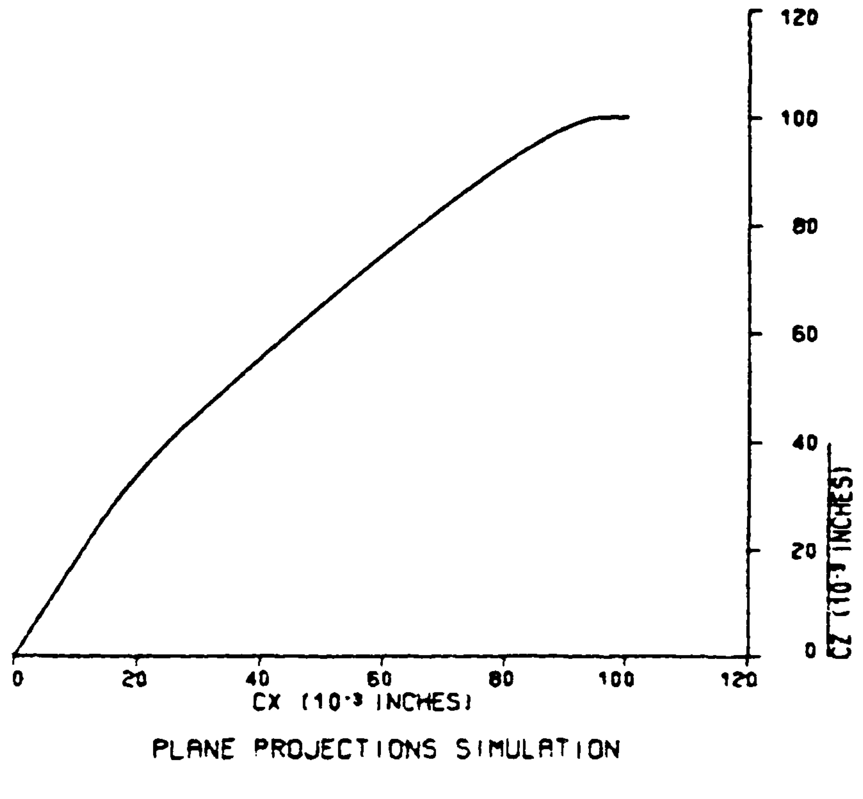


Figure 4.1(d) 0.1 STEP Excited XZ Plane Projection.

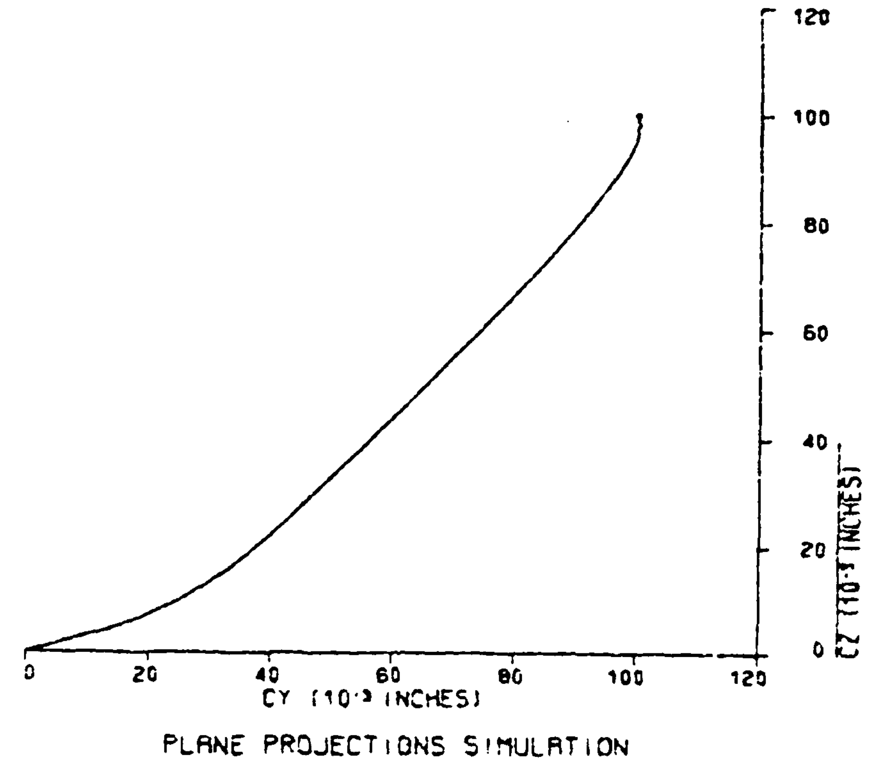


Figure 4.1(e) 0.1 STEP Excited YZ Plane Projection.

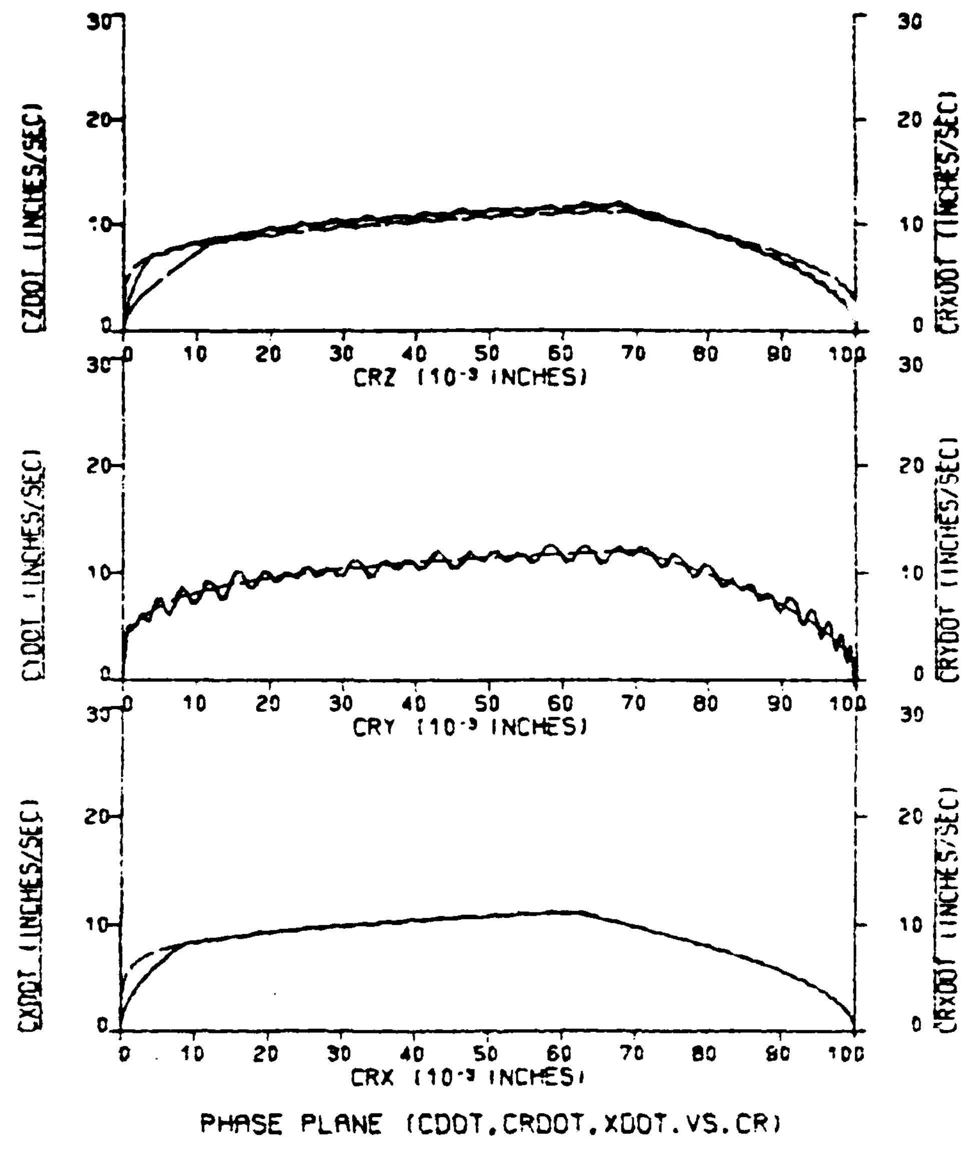


Figure 4.2(a) 0.1 RAMP Excited X, Y, and Z Step Response.
(TV1 = 0.5*FINTIME).

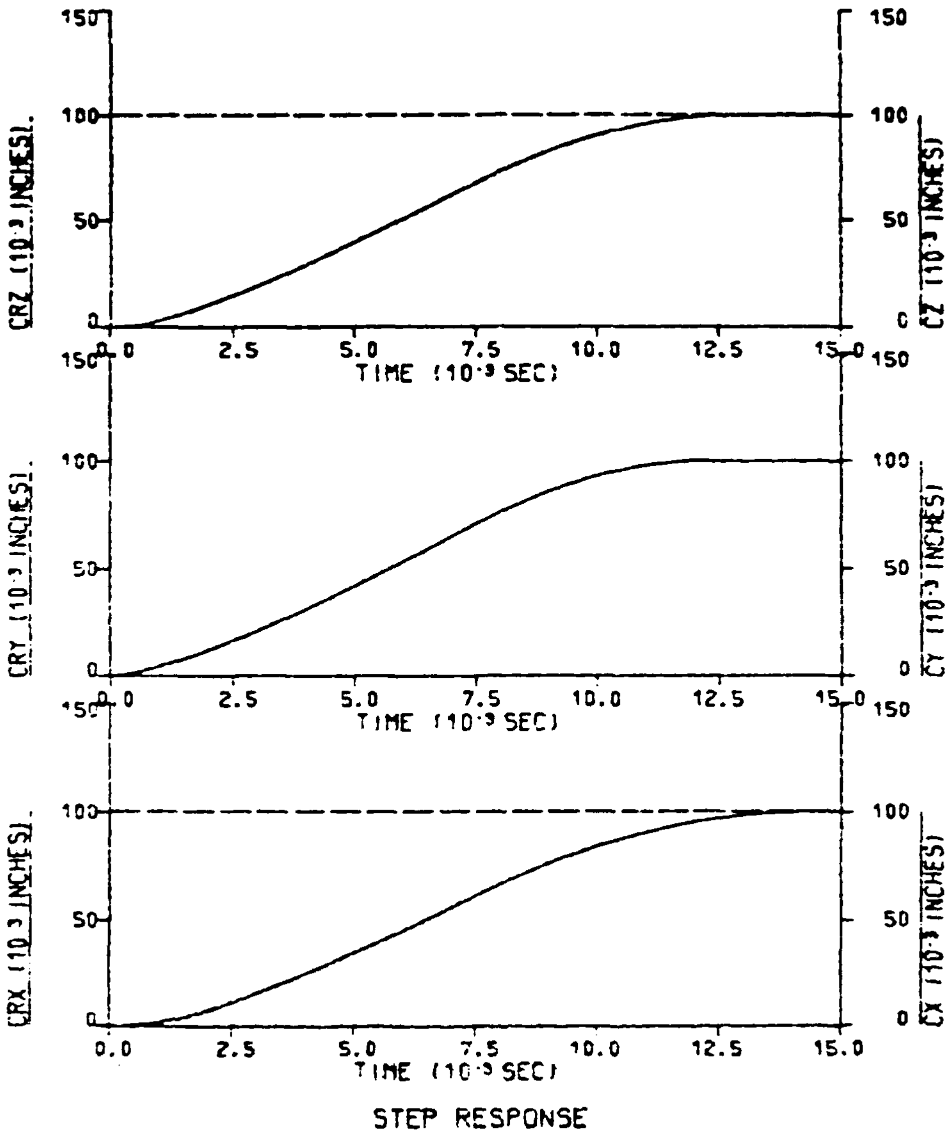
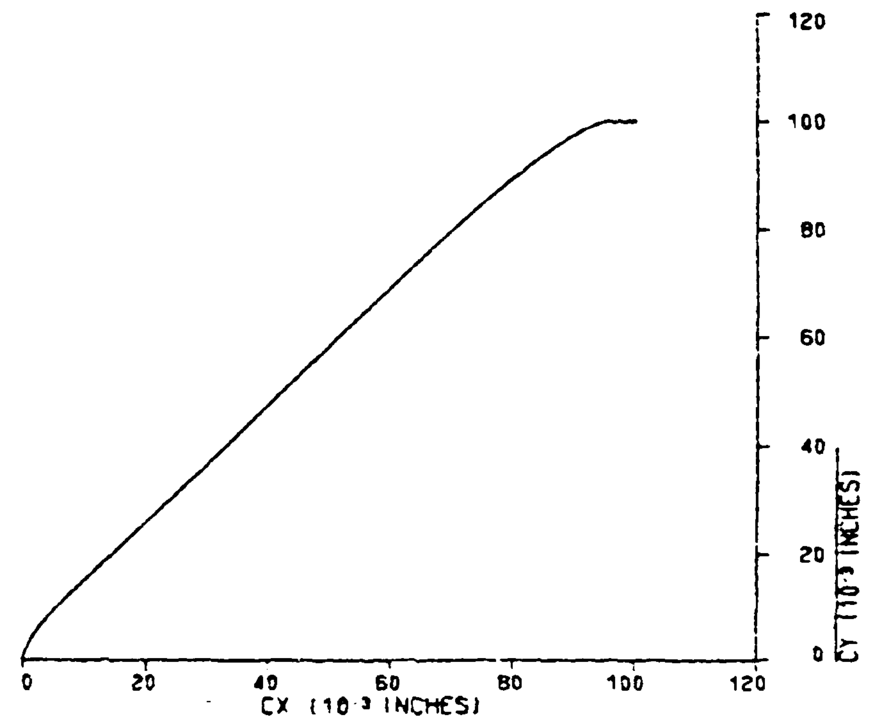


Figure 4.2(b) 0.1 RAMP Excited X, Y, and Z Phase Plane. [TV1 = 0.5*FINTIME).



PLANE PROJECTIONS SIMULATION

Figure 4.2(c) 0.1 RAMP Excited XY Plane Projection. (TV1 = 0.5*FINTIME).

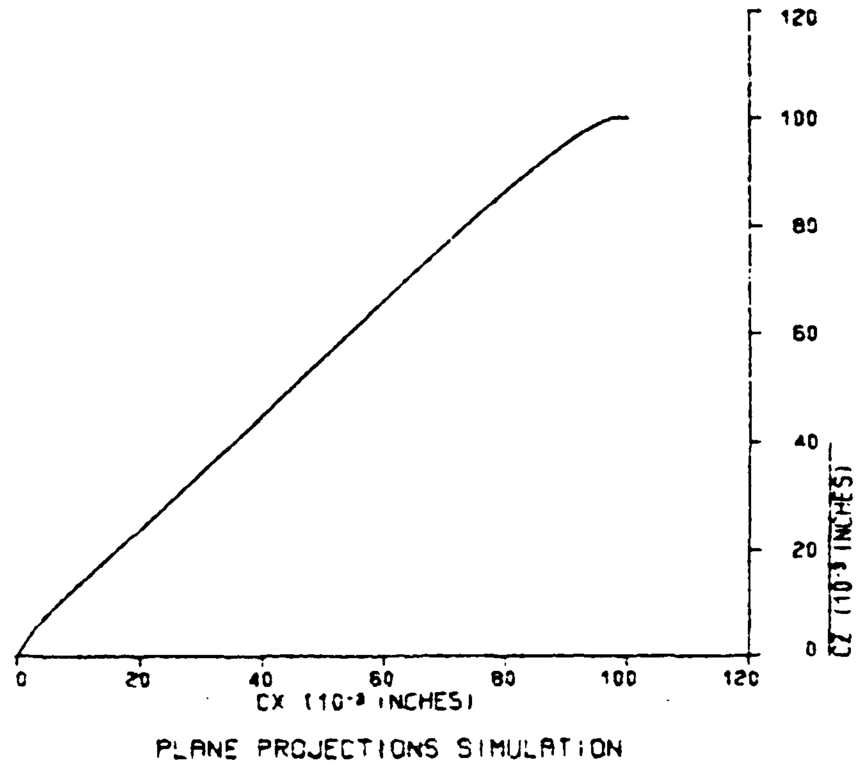


Figure 4.2(d) 0.1 RAMP Excited XZ Plane Projection. [TV1 = 0.5*FINTIME).

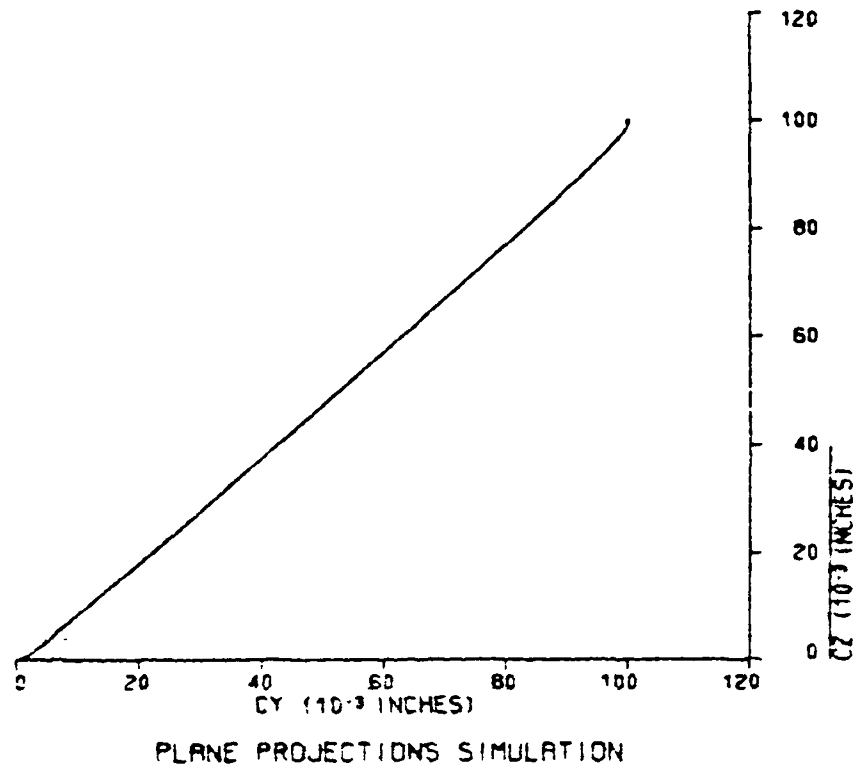


Figure 4.2(e) 0.1 RAMP Excited YZ Plane Projection. (TV1 = 0.5*FINTIME).

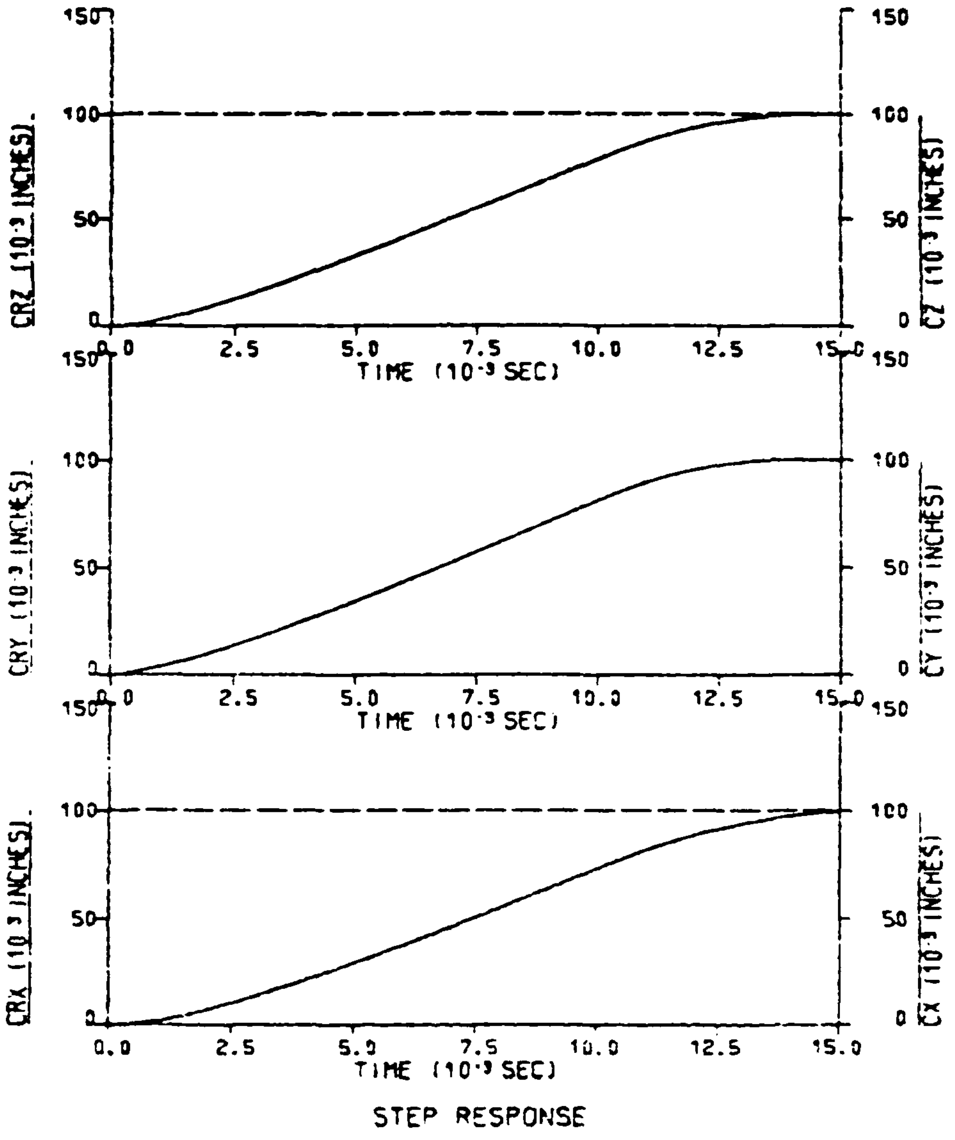


Figure 4.3(a) 0.1 RAMP Excited X, Y, and Z Step Response. (TV1 = 0.66*FINTIME).

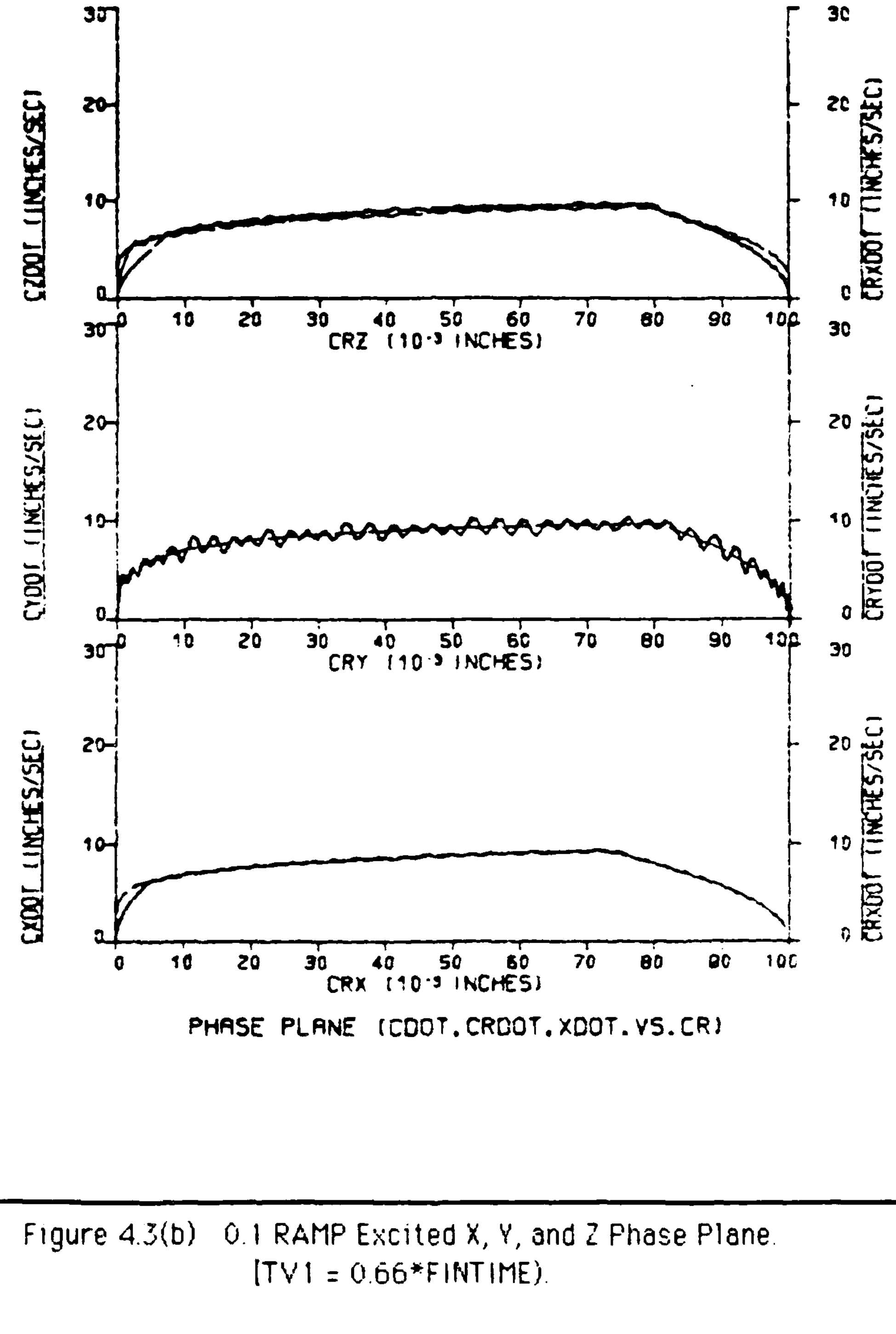


Figure 4.3(b) 0.1 RAMP Excited X, Y, and Z Phase Plane.
(TV1 = 0.66*FINTIME).

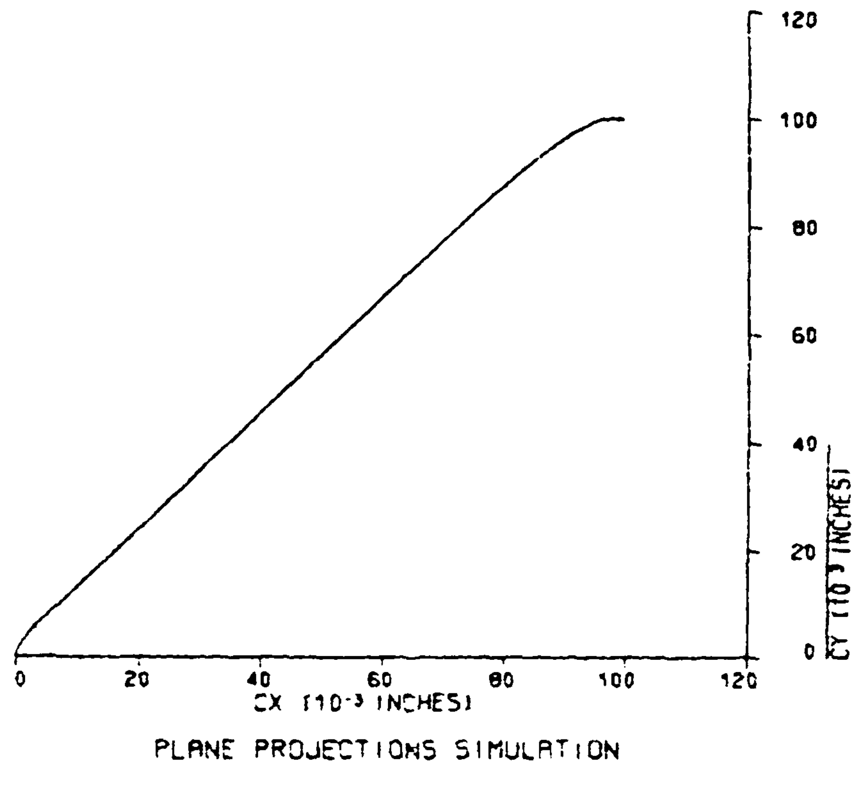


Figure 4.3(c) 0.1 RAMP Excited XY Plane Projection.
(TV1 = 0.66*FINTIME).

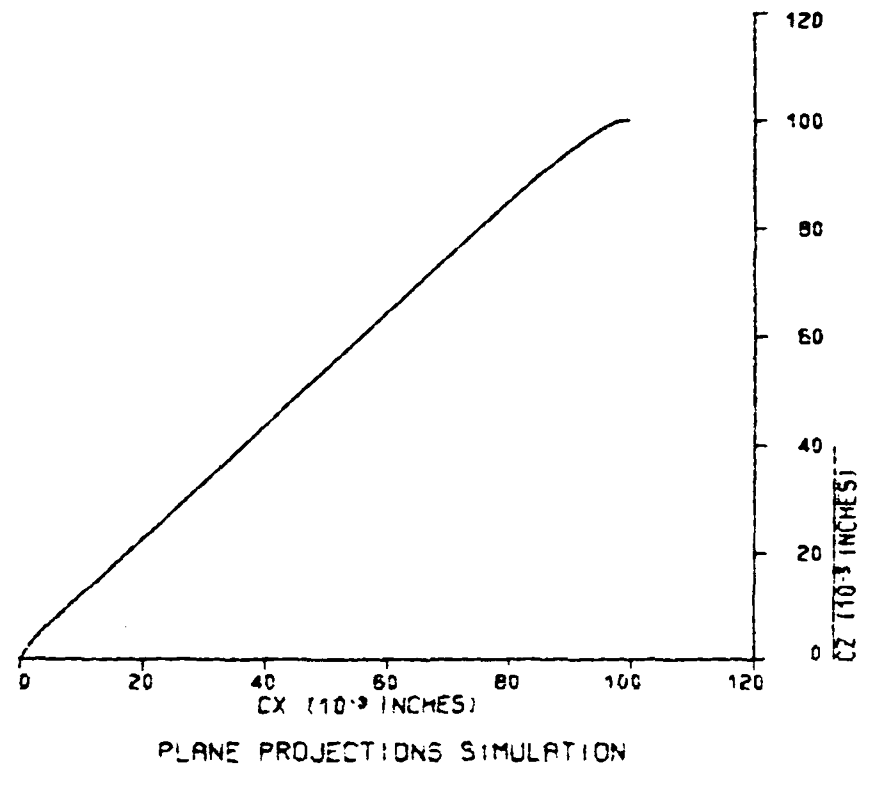
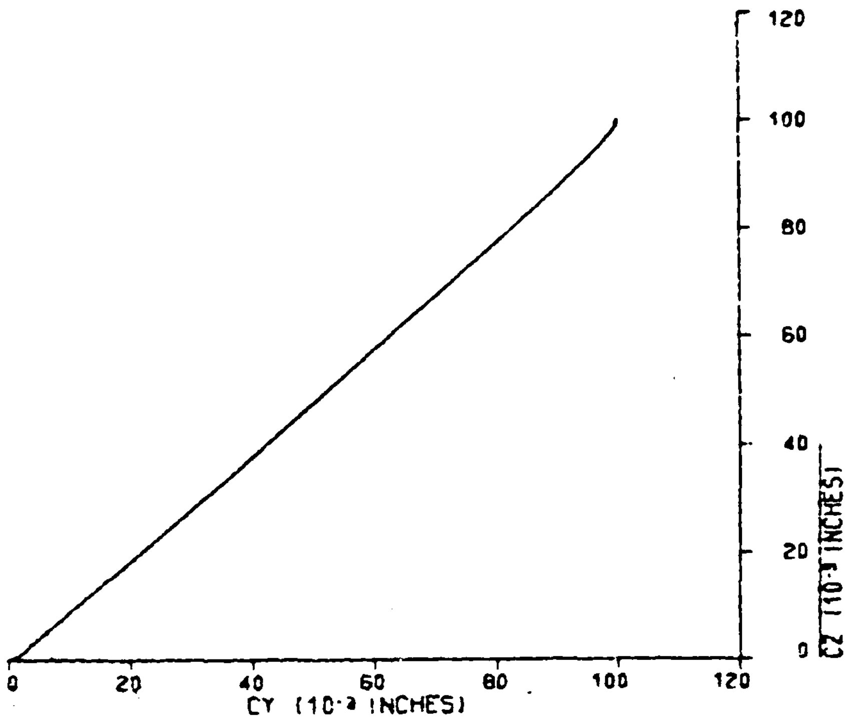
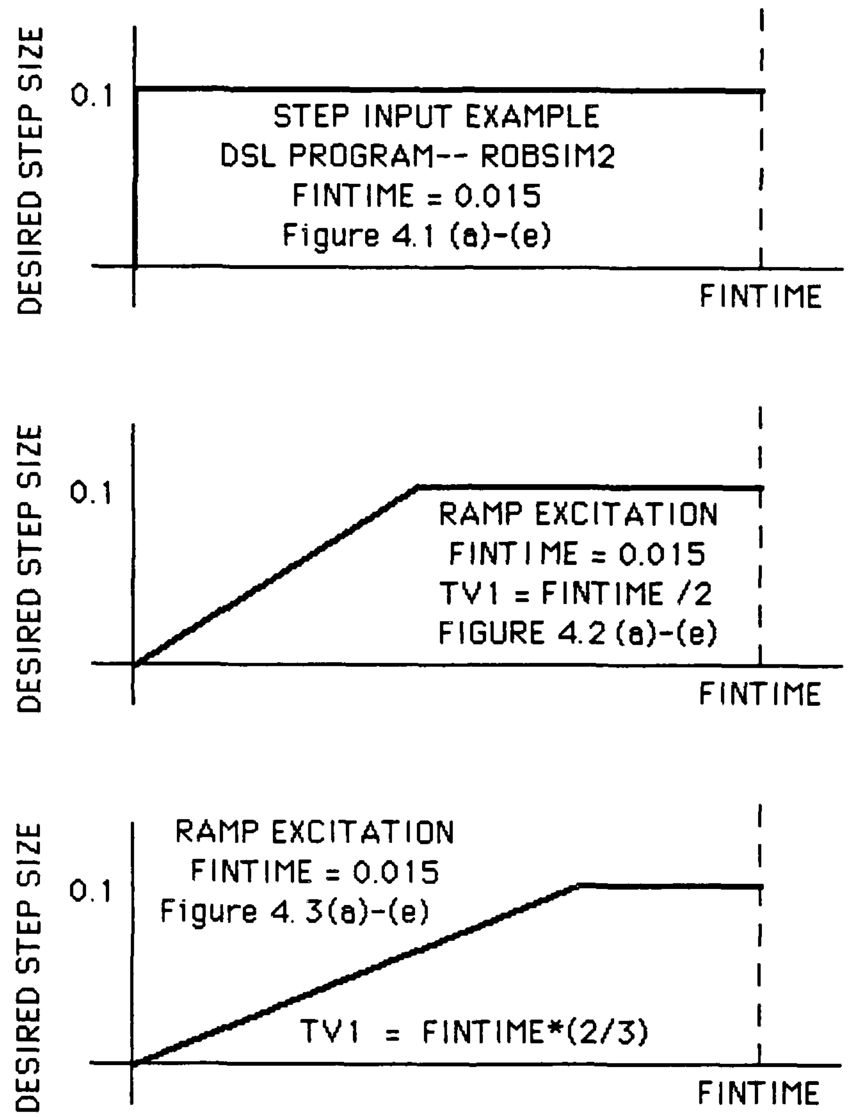


Figure 4.3(d) 0.1 RAMP Excited XZ Plane Projection. (TV1 = 0.66*FINTIME).



PLANE PROJECTIONS SIMULATION

Figure 4.3(e) 0.1 RAMP Excited YZ Plane Projection. (TV1 = 0.66*FINTIME).



STEP EQUATION IS: (DESIRED STEP SIZE)(STEP(0.0))

RAMP EQUATION IS : $\frac{\text{(DESIRED STEP SIZE)}}{\text{(TV1)}}$(TIME -RAMP(TV1))

Figure 4.4 Excitation Examples of STEP and RAMP Excitation

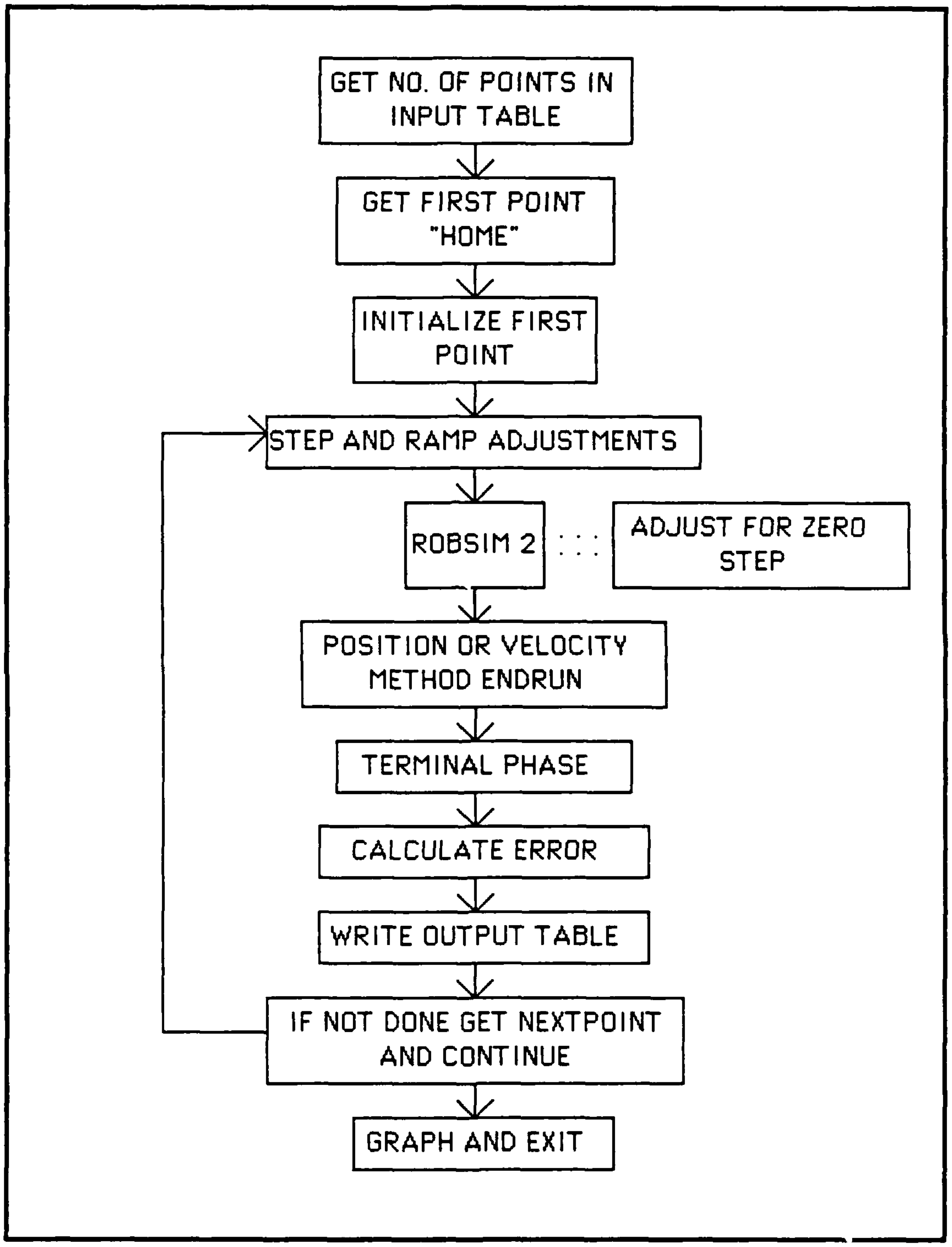


Figure 4.5 Modifications to ROBSIM2 for Multiple Runs

```
ROBSIM 1
          CONTRL
                FINTIME    =    0.06
                DELT       =    0.00005
                DELS       =    0.00025
          SAVE AT 0.0004

ROBSIM  2 AND 2N
          CONTRL
                FINTIME    =    0.015
                DELT       =    0.00001
                DELS       =    0.00005
          SAVE AT   0.00008

ROBSIM  3
          CONTRL
                FINTIME    =    0.005
                DELT       =    0.0000025
                DELS       =    0.00001
          SAVE AT    0.00002
```

The only difference in the three simulation programs are the CONTROL parameters to the DSL program and how the graphs were plotted on the TEK618. These programs were used to decide on the values that would best suit the multiple runs that are to be done.

Figure 4.6 Modifications to ROBSIM1, 2, 2n, and 3

V. COMPUTER PROGRAM PATH DEVELOPMENT

A. INTRODUCTION

Several techniques have been developed to teach robots to perform the desired tasks and movements. The most significant of these are 1.) Manual teaching, 2.) Lead-through teaching, and 3.) Programming. All of these methods require the operator to manually lead the robot through the desired task and then playback the resulting stored data or to develop sophisticated task level programs that interface with the real world using position and velocity sensors. As stated earlier the intent of this thesis is to develop a method of predetermining the control signals for each of the robot arms with an accuracy that eliminates the need for elaborate sensing equipment. Using a highly accurate positioning servo control, signals can be applied based on position alone which allows the operator to precalculate the entire path that the robot arm is to follow.

B. MANUAL TEACHING and LEAD-THROUGH TEACHING

Sometimes called teaching-by-showing or guiding involves manipulating the robot arms through the desired patterns, storing the resulting time and position coordinates in memory, and playing back the coordinates in the desired sequence. The Manual Teaching method does not require a computer to control the movements and is used mainly in repetitive tasks such as welding, painting, or material handling. The Lead-Through requires a computer or simulator to control. Disadvantages to both methods include large memory requirements, unintentional motions will be recorded as the arm is manually lead through the task, high precision is not possible, and the exact velocity of the arm cannot be controlled.

C. PROGRAMMING TECHNIQUES

Programming of complex or random maneuvers allows the robot arm to be accurately positioned. Robot level programming correlates the robots sensor data and then specifies the desired movement. Motion by the robots arm can be predetermined by either of the teach methods described earlier and uses sensors to tell the position of an object and bases the motion of the robot relative to the coordinate system of the object. With Task level programming the user describes the task and the task-planner formulates a robot level program to carry out that task. Either of the level programming techniques require a special language to be used in controlling the robot.

D. COMPUTER PATH DEVELOPMENT

Instead of developing the coordinates for the robot movement by the teach methods we are precalculating the required robot control signals and then applying these control signals to each of the arms in an independent but simultaneous manner. First the desired path that the robot is to follow is described and then these values are adjusted for the type of robot arm positioning syncros being used that were described in Chapter Three. Finally, in Chapter Six, the desired control signals are applied to the simulated model and the motion of the simulator is compared to the desired path.

1. <u>Desired Path</u>

Options in picking the desired path have been limited to make the program manageable. The following is a list of the assumptions and restrictions in picking the path:

1. Paths are of the form Y = f(X) and Z = f(X,Y).
2. Start and finish points are inputted as two-tuples of (X,Y). Start and finish points in the Z-direction are calculated from the equations that are implemented in the program. Motion in just one direction can not be accomplished in the cartesian coordinate system because of the difficulty in describing the motion. A parametric routine, added to make conic section equations easier to program, will be able to easily handle these types of equations.
3. All values must result in positive values that will be weighted as desired by the user. This weight factor is a linearization so that the path coordinates are within the range of the robot. This is termed scaling and will be discussed further in Chapter Six.
4. Maximum travel lengths are based on the maximum velocity of the separate arms and by the differences in each of the arm torques. The current programs assume that the velocities are equal to the maximum step length divided by a fixed delta T and are scaled so that the maximum step sizes are all equal in all three coordinate directions.
5. The user has a choice of going home after each movement or going to the start point of the next equation. If the start of the next equation is the same as the finish of the previous you must do a FSTAR

subroutine to align the index variables. This inserts a repeated step where the robot goes nowhere and does not increase delay time significantly. The cost to the DSL program is just a few instructions before the program goes and gets the next point.

6. Three equation capability was chosen for path description in this system for simplicity, yet being sufficiently long enough to demonstrate the required path structure (ROBPATH1). A two equation capability will be discussed as the primary program later in this chapter (ROBPATH4).
7. The program has been limited to 1000 points of (X,Y,Z) coordinate points and a maximum distance for the each movement has been chosen to be 0.10 units to allow each of the points to be discernible. To observe smoothness a maximum distance for movement will be reduced to 0.01 or 0.001 depending on the application. These will be demonstrated further in Chapter Six.
8. Movement in a plane parallel to the axis has been accounted for.
9. Due to the method of scaling the start and finish points of each arm must be different and distinct relative to the X coordinate axis. That is the start and finish points must be in the direction of the slope of the desired curve. This determines the direction that the X increment is taking so that when applied to y=f(x) we get a legitimate y increment.

2. Cartesian Robot Motion

Since the robot of concern here is a cartesian robot the allowable motion has been restricted to the first octant where all of the (X,Y,Z) coordinate points are positive and the coordinate system is right-handed. This means that the program will not be sufficient if used with non-cartesian robots without some modifications or transformations. Transformations will be discussed further in Chapter Seven. Figure 5.1 gives a description of the areas that are required to be modified for successful run of the program. Figure 5.2 flowcharts the input/change procedure required to make the program function. The full program listing in both FORTRAN [WF77] and Microsoft BASIC are located in Appendix E and F. Figures 5.3 to 5.7 flowchart the program main and the four subroutines used in the ROBPATH4.FORTRAN program. Flowcharts for the BASIC program are given in Appendix F. From now on we will refer to the ROBPATH4.FORTRAN listing in all explanations.

To set up the program for defining the desired path, or series of sub-paths, of the robot tip (sometimes called the endpoint) follow the procedure set up in Figure 5.2 and as seen in example on Figure 5.1. The user must first decide on the motion that he wants the robot tip to make following the restrictions stated earlier. Once the path has been chosen then the user can modify the FORTRAN program so that the desired calculations will follow the path. There are four main subroutines which help in formulating the desired path movement. They are:

1. **HSTAR**: Describes motion of the Robot Arm from Home to the Start of the next desired path. Motion is linear between Home and Start points.

2. **STARF**: Describes the motion of a path from the Start to Finish of the desired sub-path. This is the program that calculates a set of three tuples of position points based on user inputted equations in Cartesian Coordinates.
3. **FSTAR**: Describes the motion of a path from the Finish of one sub-path to the Start of the next sub-path. Motion is linear in this subroutine.
4. **FHOME**: Describes the motion of a path from the Finish of a sub-path to Home. Motion is linear in this subroutine.
5. **SFPAR**: Describes the motion of a path from Start to Finish of a set of X, Y, and Z parametric equations which are a function of a single variable. The variable can represent radian movement, time movement, or can represent a user defined variable. Start and Finish points are user supplied, positive, with the start point [TS] having a smaller magnitude then the finish [TF]. User must take this into consideration when writing the parametric equations.

All of the subroutines are similar in construction except STARF and SFPAR. They assume that the path between the start and finish points are linear and use these points to scale the all of the incremental values to within the maximum distance that each coordinate can move. With the subroutine STARF the main assumption is that changes in X [DELX] determine the value of the change in Y [DELY] and that these in turn determine the incremental change in Z [DELZ]. First the value of the difference in the X start and finish directions are used to determine the direction of the x increment. Then this value is applied to the first equation [EQAT1] to determine the corresponding y increment. If this increment is greater than the maximum value allowed in this direction [MAXODY] then the x increment is linearly scaled using the following relationship:

$$\text{DELX} = \text{DELX (MAXODY/ABS(DELY))}. \qquad \text{[eqn 5.1]}$$

This scaling is repeated until both values are within their prescribed maximum values. Then the x and y increments are used to calculate the z increment [DELZ] using the second equation [EQUAT1]. If the z increment is larger than the maximum allowed z increment [MAXODZ] then all three increments are scaled to make the z increment equal to MAXODZ. Then all three increments are added to the previously calculated three tuple of coordinate positions [HSPOS array] and the process is repeated for the next set of points. Each time a point is generated there is a check to see if the path (or sub-path) has reached the FINISH point. If the FINISH point has been reached then the last position value is set equal to the FINISH point and the program continues on to the next subroutine. The program also checks for plane movement and if plane movement is noted then the value of the previous data point is retained.

As each of the subroutines are called the HSPOS array maintaining the data is read into the program and then rewritten. These sets of data points are to be used in driving the DSL Simulation program [chosen in Chapter Four] and to compare the final results. The main array is HSPOS which is composed of a the total number of three tuples in the array followed by columns containing the three tuple of coordinate positions which will then be used by one of the DSL simulation programs.

Chapter Six takes the results of the FORTRAN program, which are the desired path coordinates, and compares each point with the results out of the DSL program. It will also contain any modifications to the programs if they are needed, such as scaling.

- USER MODIFICATION REGION
  - FUNCTION MODIFICATION
    - EQUATION 1-4 (EQAT 1-4 /EQUAT1-4)
    - PARAMETRIC X/Y/Z EQUATION 1-4
  - MAIN PROGRAM
    - DELT
    - MAXOD
    - ACCURACY (IF USED)(ACC)
    - MAX VELOCITY CONSTANTS (CURRENTLY 1)
    - DELT (AGAIN)
    - VALUE MAXIMUM (VALMAX)
    - VALUE MIINIMUM (VALMIN)
  - EQUATIONS BOUNDS AREA
    - START AND FINISH POINTS
    - OUTPUT S/F POINTS
  - PATH MOVEMENT REGION
    - HOME (DIFFERENT FOR DIFFERENT ARM CONFIGURATIONS AND TYPE)
    - PATH FORMATION
  - PRINTOUT REGION
    - ROBOT SIMULATION NUMBER

CHECKING THESE PARAMETERS ALONG WITH FIGURE 5.2 WILL GIVE YOU THE PATH. AFTER UPDATING THE FILDEFS NEEDED AND COMPILING IT IS RECOMMENDED THAT A THREE DIMENSION PLOT BE DONE TO INSURE THAT THE PATH IS AS DESIRED.

Figure 5.1 Path Program Modification Regions.

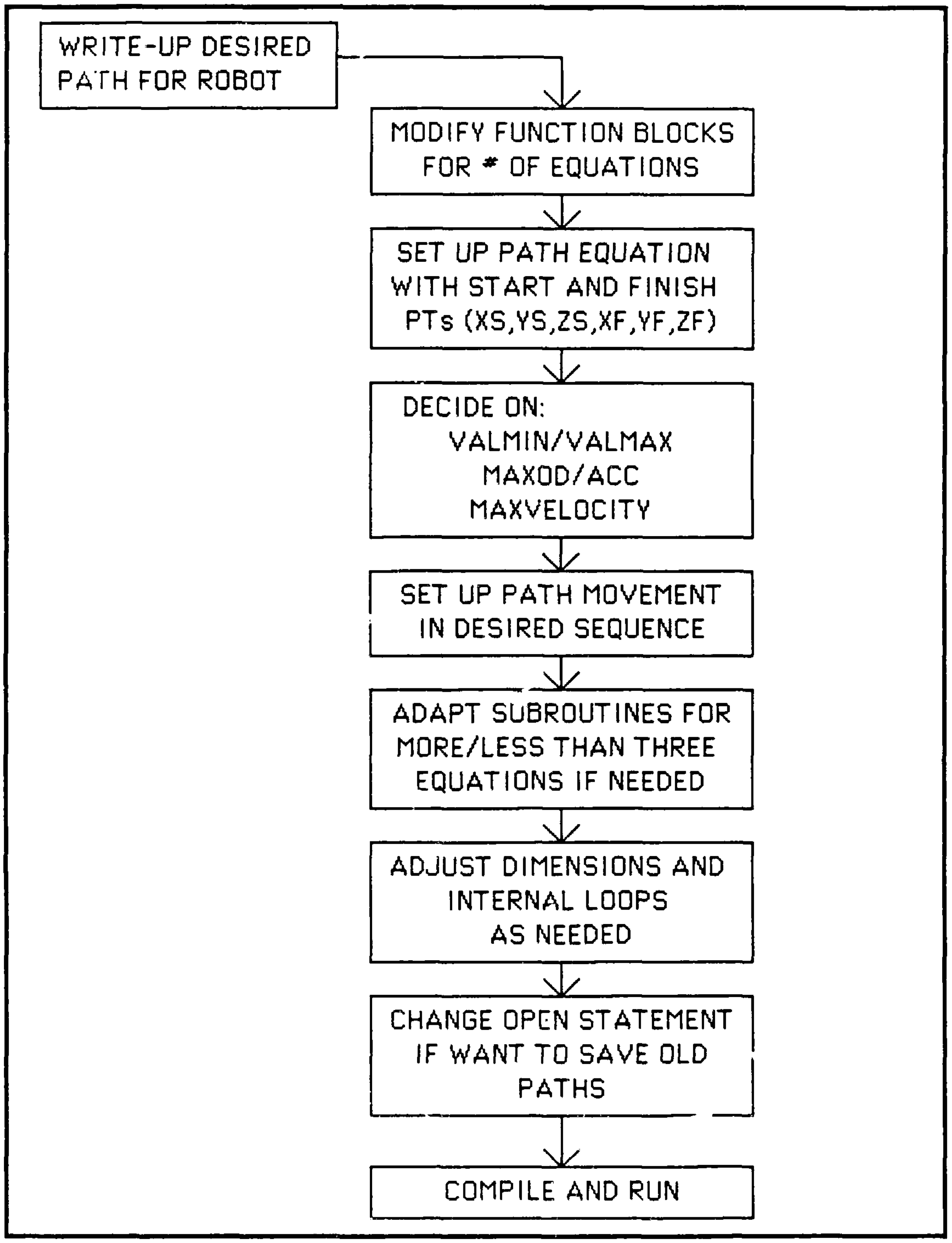

WRITE-UP DESIRED PATH FOR ROBOT
MODIFY FUNCTION BLOCKS FOR # OF EQUATIONS
SET UP PATH EQUATION WITH START AND FINISH PTs (XS,YS,ZS,XF,YF,ZF)
DECIDE ON: VALMIN/VALMAX MAXOD/ACC MAXVELOCITY
SET UP PATH MOVEMENT IN DESIRED SEQUENCE
ADAPT SUBROUTINES FOR MORE/LESS THAN THREE EQUATIONS IF NEEDED
ADJUST DIMENSIONS AND INTERNAL LOOPS AS NEEDED
CHANGE OPEN STATEMENT IF WANT TO SAVE OLD PATHS
COMPILE AND RUN

Figure 5.2 Procedure for Updating Path Equation.

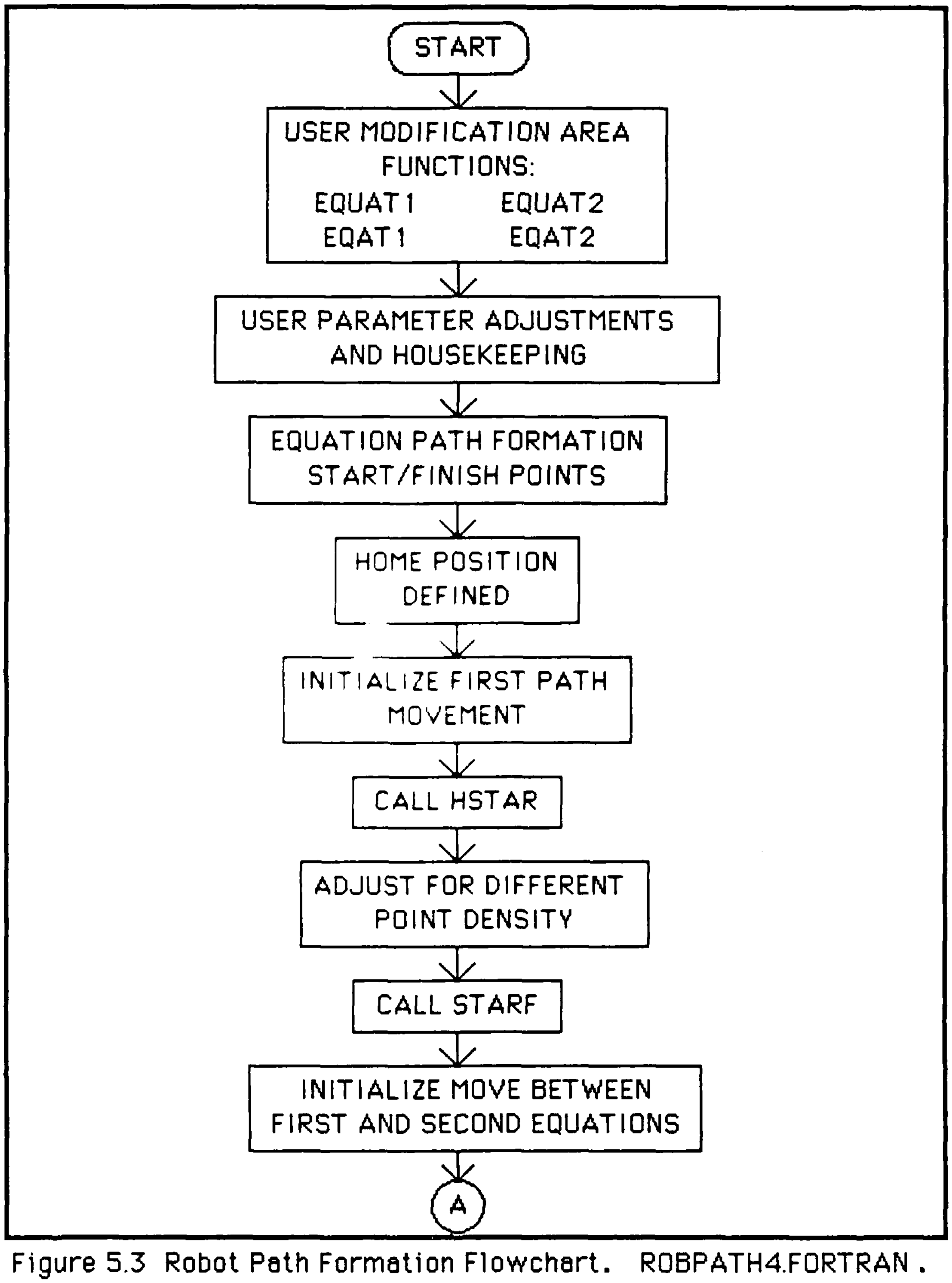
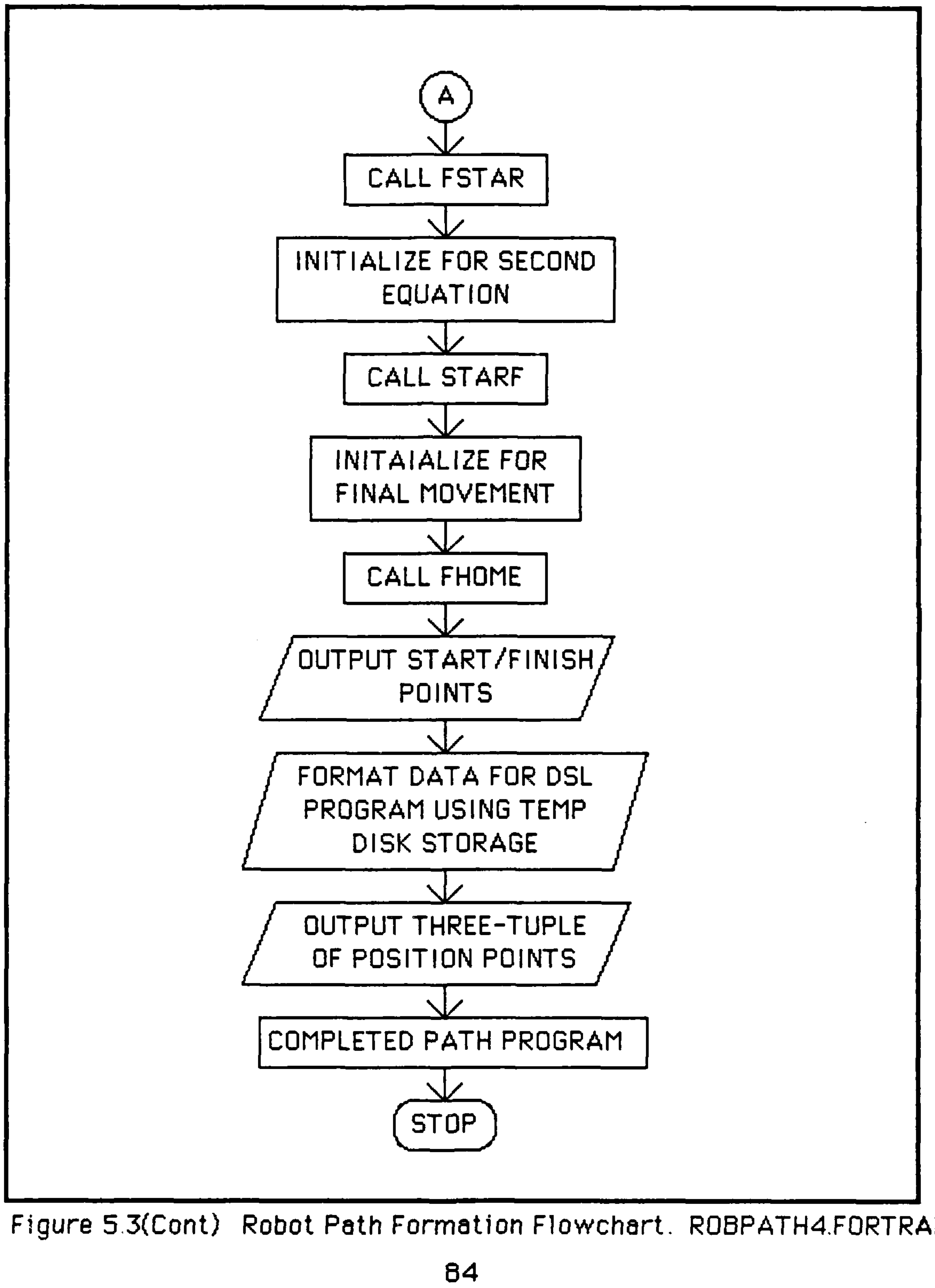

START
USER MODIFICATION AREA FUNCTIONS:
EQUAT1 EQUAT2
EQAT1 EQAT2
USER PARAMETER ADJUSTMENTS AND HOUSEKEEPING
EQUATION PATH FORMATION START/FINISH POINTS
HOME POSITION DEFINED
INITIALIZE FIRST PATH MOVEMENT
CALL HSTAR
ADJUST FOR DIFFERENT POINT DENSITY
CALL STARF
INITIALIZE MOVE BETWEEN FIRST AND SECOND EQUATIONS
A

Figure 5.3 Robot Path Formation Flowchart. ROBPATH4.FORTRAN .


A
CALL FSTAR
INITIALIZE FOR SECOND EQUATION
CALL STARF
INITAIALIZE FOR FINAL MOVEMENT
CALL FHOME
OUTPUT START/FINISH POINTS
FORMAT DATA FOR DSL PROGRAM USING TEMP DISK STORAGE
OUTPUT THREE-TUPLE OF POSITION POINTS
COMPLETED PATH PROGRAM
STOP
Figure 5.3(Cont) Robot Path Formation Flowchart. ROBPATH4.FORTRAN

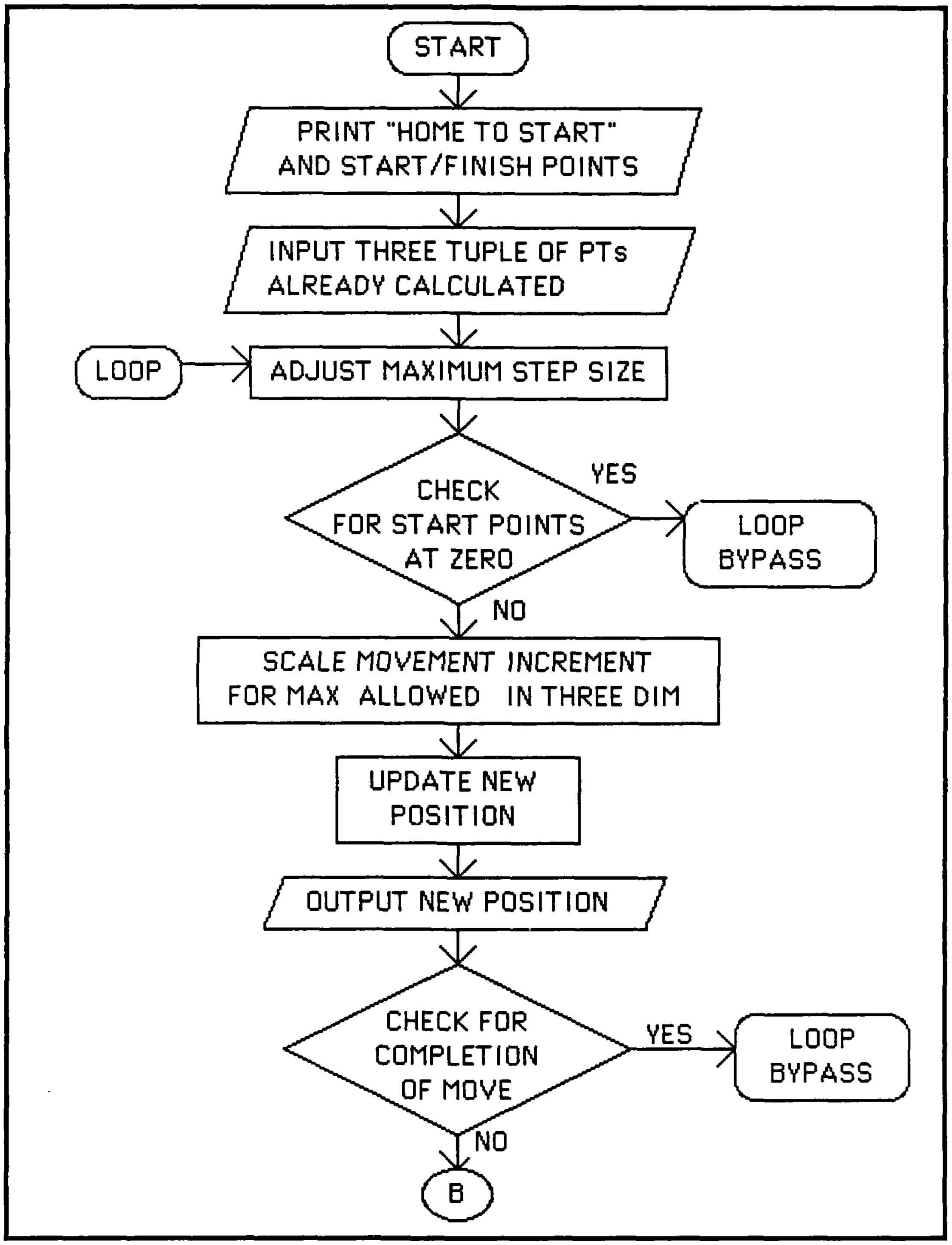

START
PRINT "HOME TO START" AND START/FINISH POINTS
INPUT THREE TUPLE OF PTs ALREADY CALCULATED
LOOP
ADJUST MAXIMUM STEP SIZE
CHECK FOR START POINTS AT ZERO
YES
LOOP BYPASS
NO
SCALE MOVEMENT INCREMENT FOR MAX ALLOWED IN THREE DIM
UPDATE NEW POSITION
OUTPUT NEW POSITION
CHECK FOR COMPLETION OF MOVE
YES
LOOP BYPASS
NO
B

Figure 5.4 Subroutine HSTAR. Home to Start.

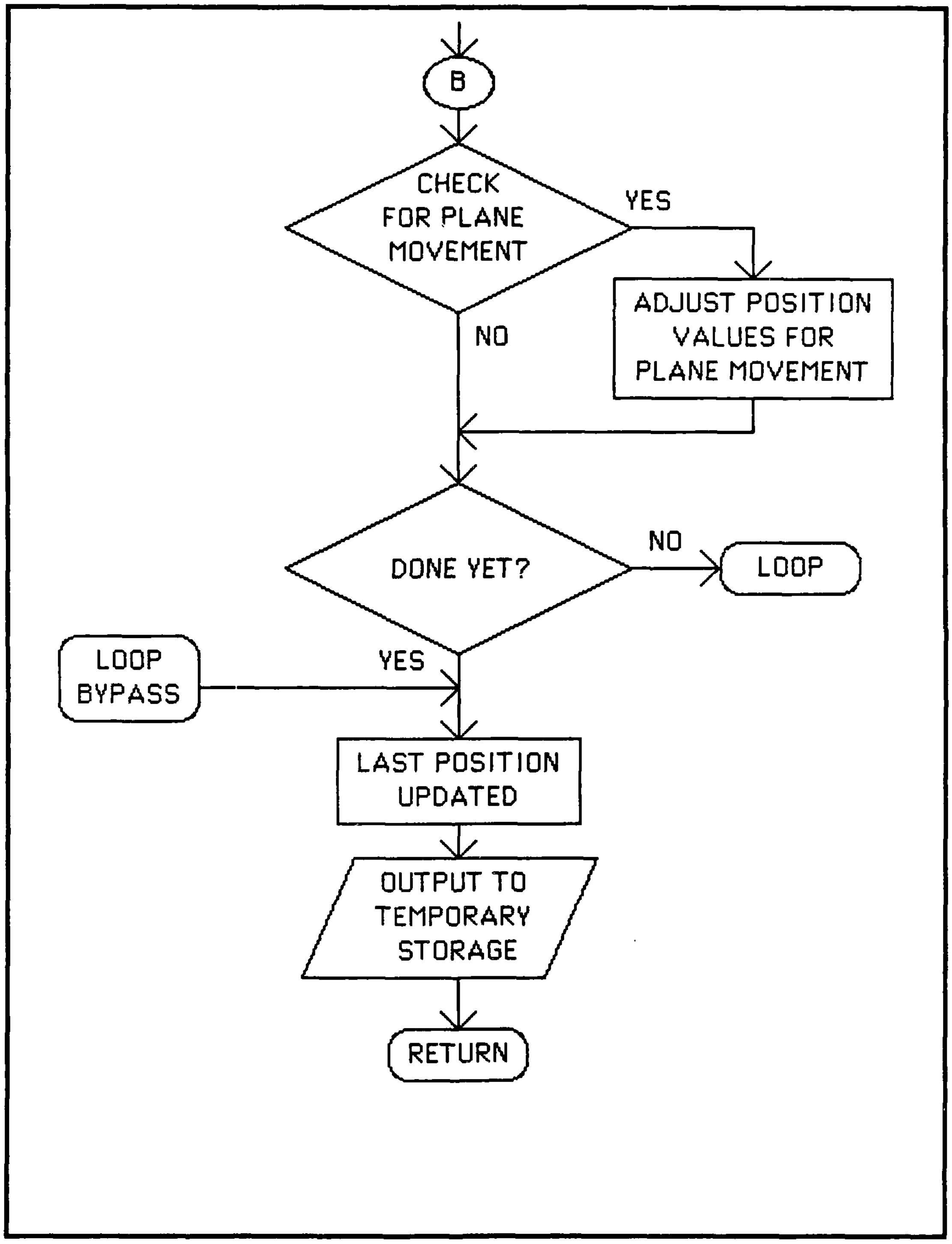


Figure 5.4 (Cont) Subroutine HSTAR. Home to Start.

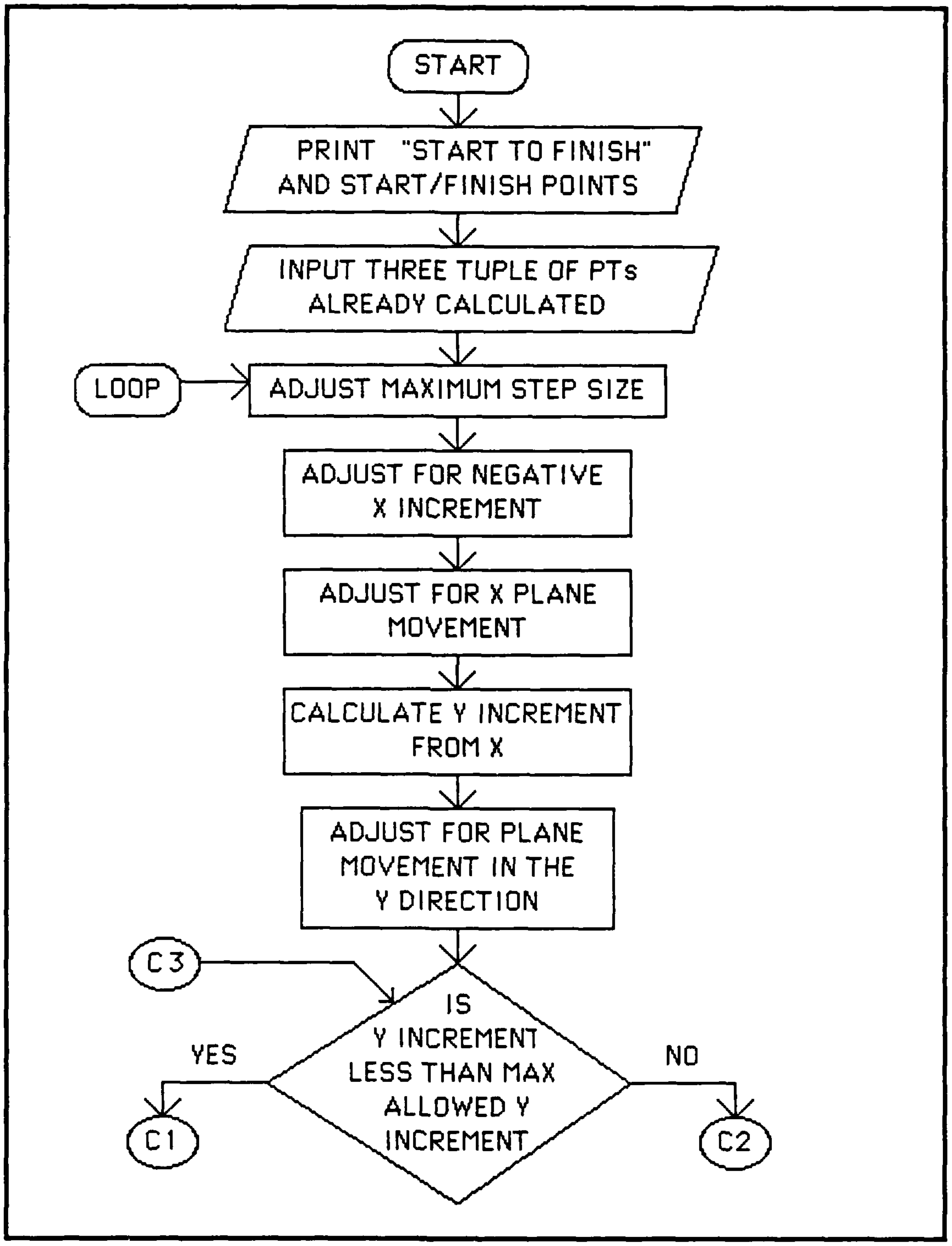

START
PRINT "START TO FINISH" AND START/FINISH POINTS
INPUT THREE TUPLE OF PTs ALREADY CALCULATED
LOOP
ADJUST MAXIMUM STEP SIZE
ADJUST FOR NEGATIVE X INCREMENT
ADJUST FOR X PLANE MOVEMENT
CALCULATE Y INCREMENT FROM X
ADJUST FOR PLANE MOVEMENT IN THE Y DIRECTION
C3
IS Y INCREMENT LESS THAN MAX ALLOWED Y INCREMENT
YES
NO
C1
C2

Figure 5.5 Subroutine STARF. Start to Finish.

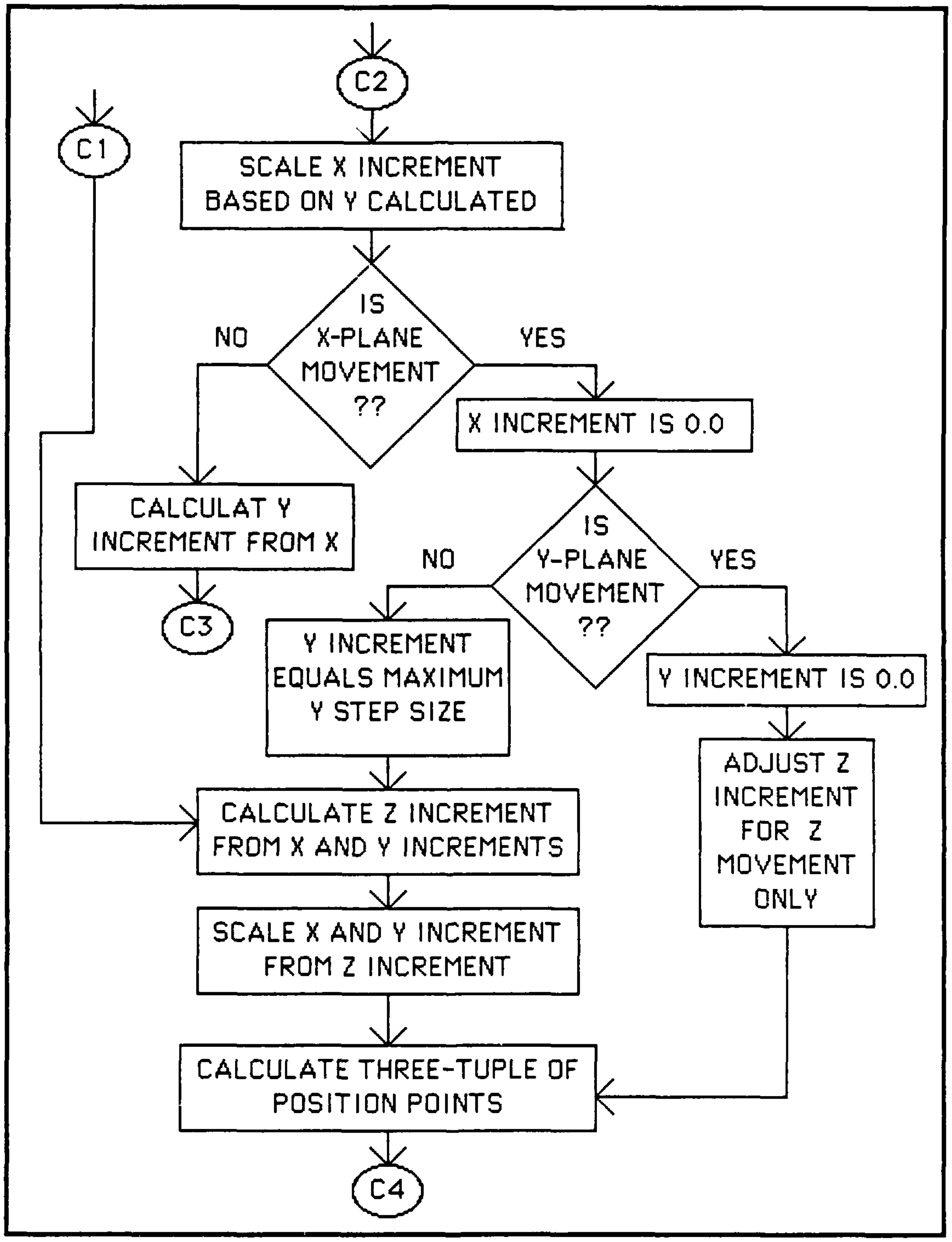

C2
C1
SCALE X INCREMENT BASED ON Y CALCULATED
IS X-PLANE MOVEMENT ??
NO
YES
X INCREMENT IS 0.0
CALCULAT Y INCREMENT FROM X
C3
IS Y-PLANE MOVEMENT ??
NO
YES
Y INCREMENT EQUALS MAXIMUM Y STEP SIZE
Y INCREMENT IS 0.0
ADJUST Z INCREMENT FOR Z MOVEMENT ONLY
CALCULATE Z INCREMENT FROM X AND Y INCREMENTS
SCALE X AND Y INCREMENT FROM Z INCREMENT
CALCULATE THREE-TUPLE OF POSITION POINTS
C4

Figure 5.5 (Cont) Subroutine STARF. Start to Finish.

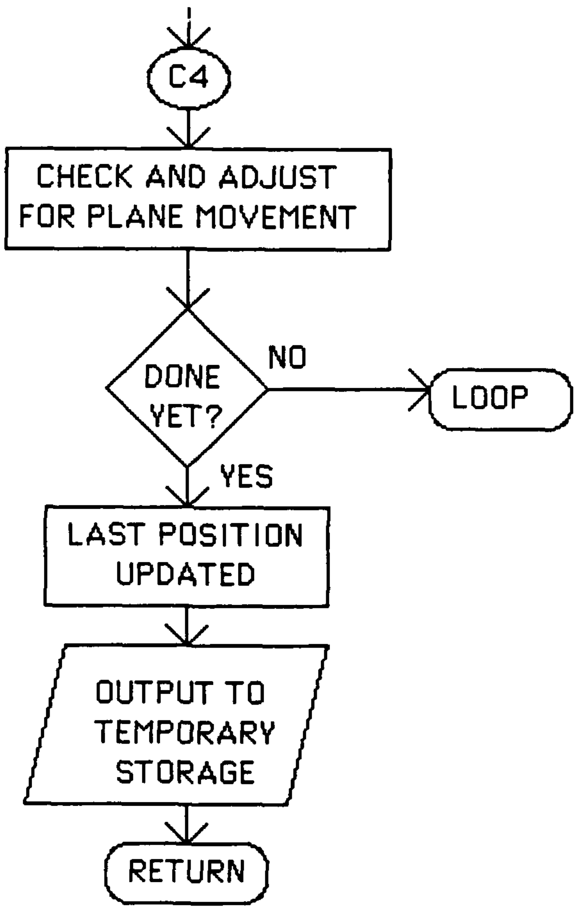

C4
CHECK AND ADJUST FOR PLANE MOVEMENT
DONE YET?
NO
LOOP
YES
LAST POSITION UPDATED
OUTPUT TO TEMPORARY STORAGE
RETURN

Figure 5.5 (Cont) Subroutine STARF. Start to Finish.

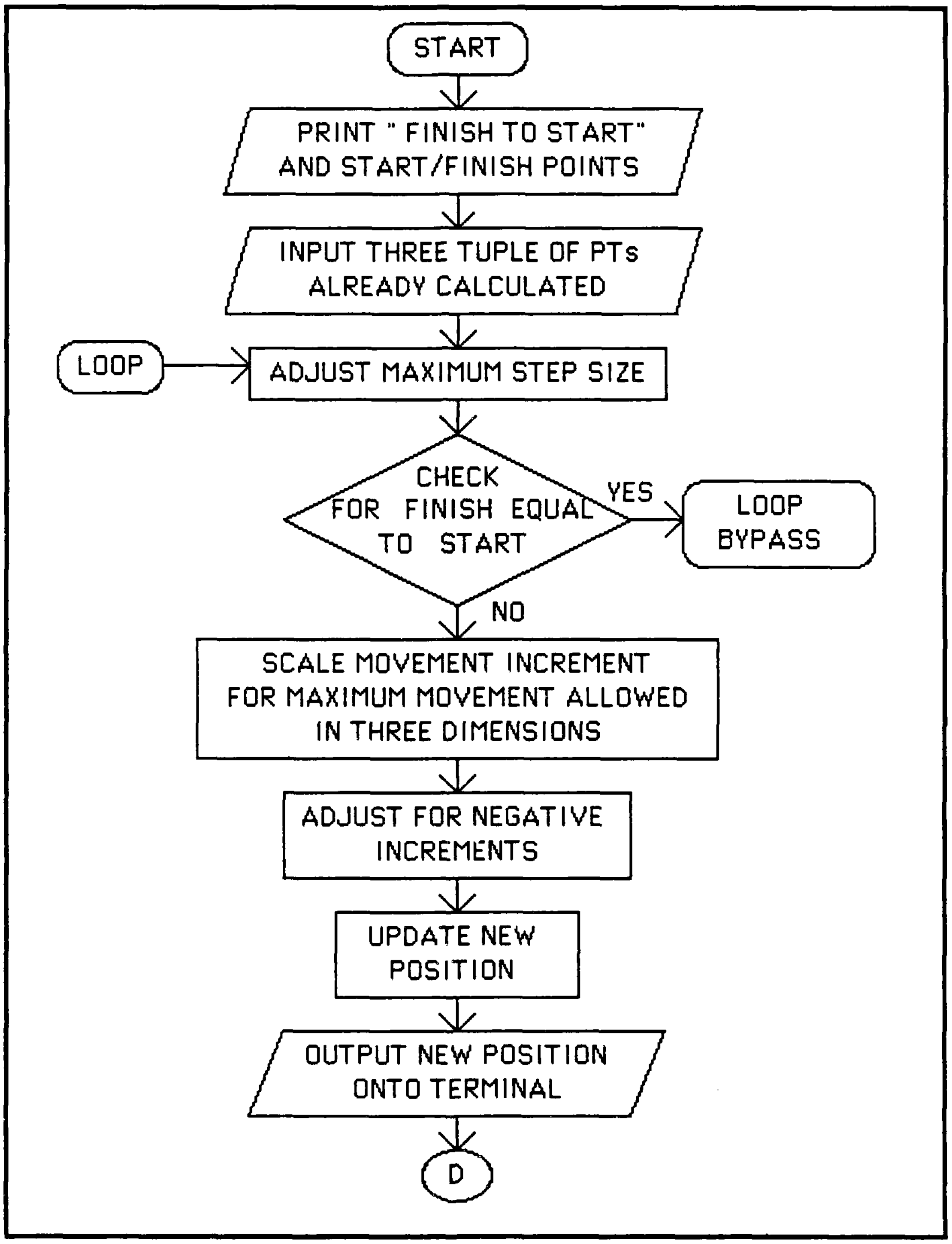


Figure 5.6 Subroutine FSTAR. Finish to Start.

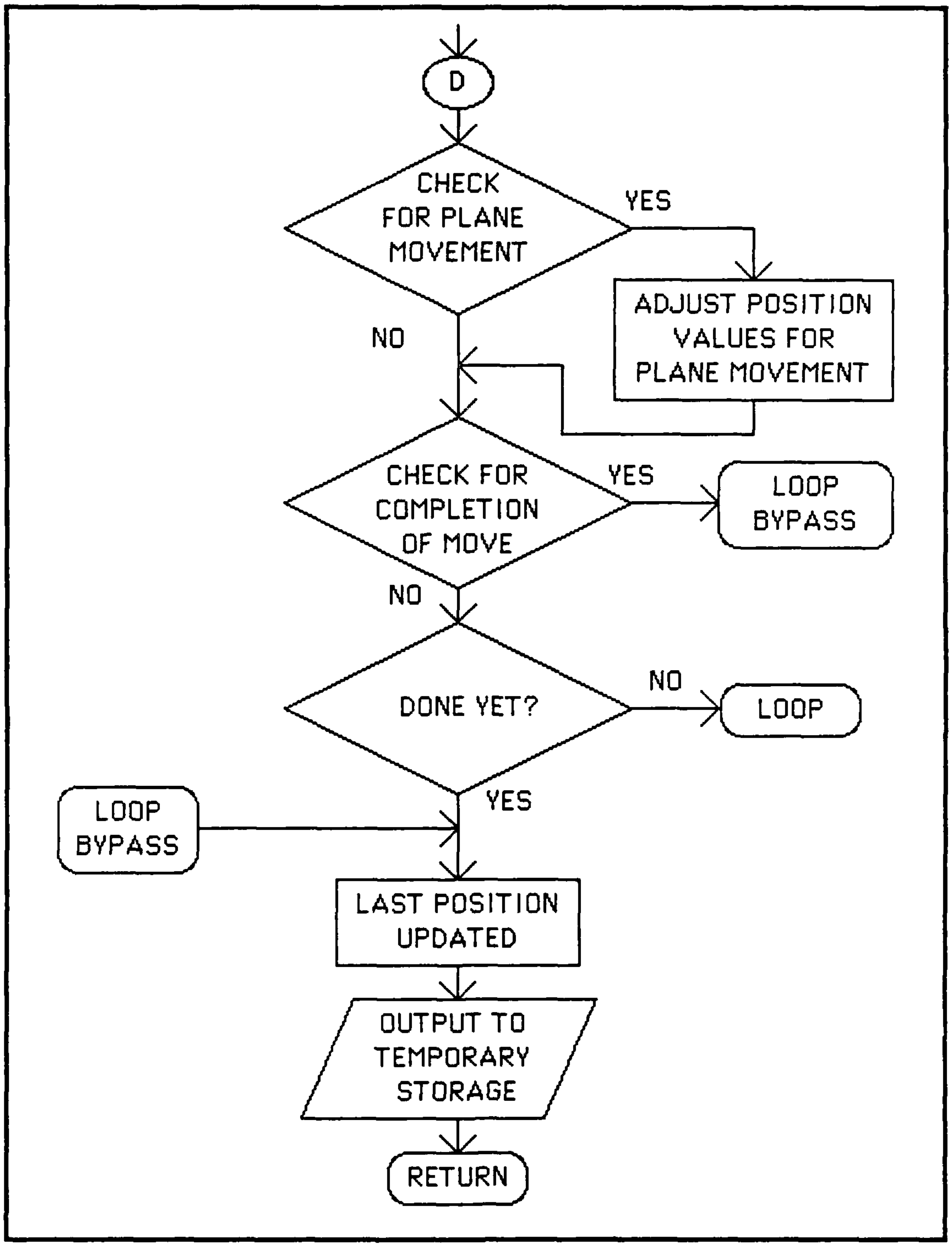


Figure 5.6 (Cont) Subroutine FSTAR. Finish to Start.

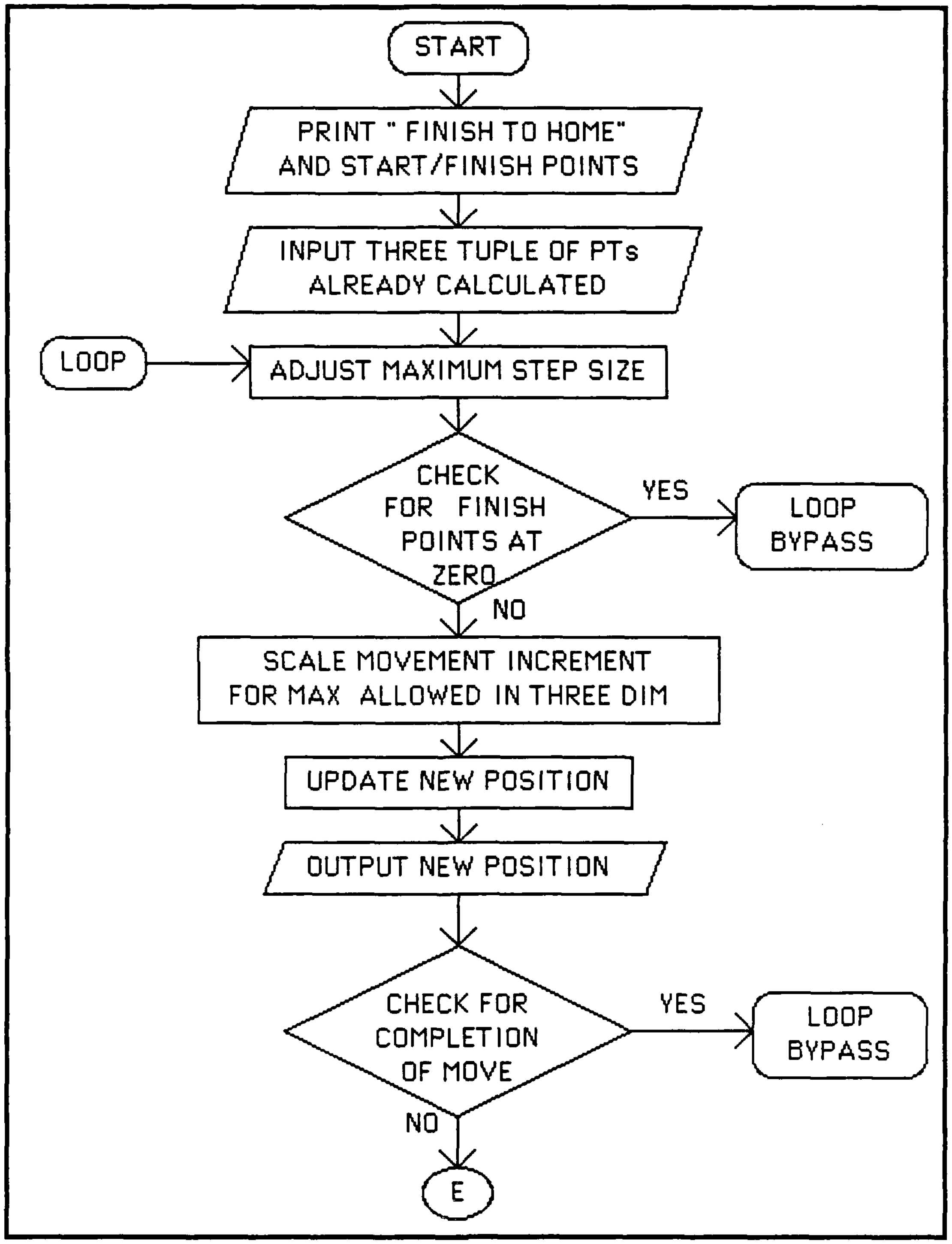

START
PRINT " FINISH TO HOME" AND START/FINISH POINTS
INPUT THREE TUPLE OF PTs ALREADY CALCULATED
LOOP
ADJUST MAXIMUM STEP SIZE
CHECK FOR FINISH POINTS AT ZERO
YES
LOOP BYPASS
NO
SCALE MOVEMENT INCREMENT FOR MAX ALLOWED IN THREE DIM
UPDATE NEW POSITION
OUTPUT NEW POSITION
CHECK FOR COMPLETION OF MOVE
YES
LOOP BYPASS
NO
E

Figure 5.7 Subroutine FHOME. Finish to Home.

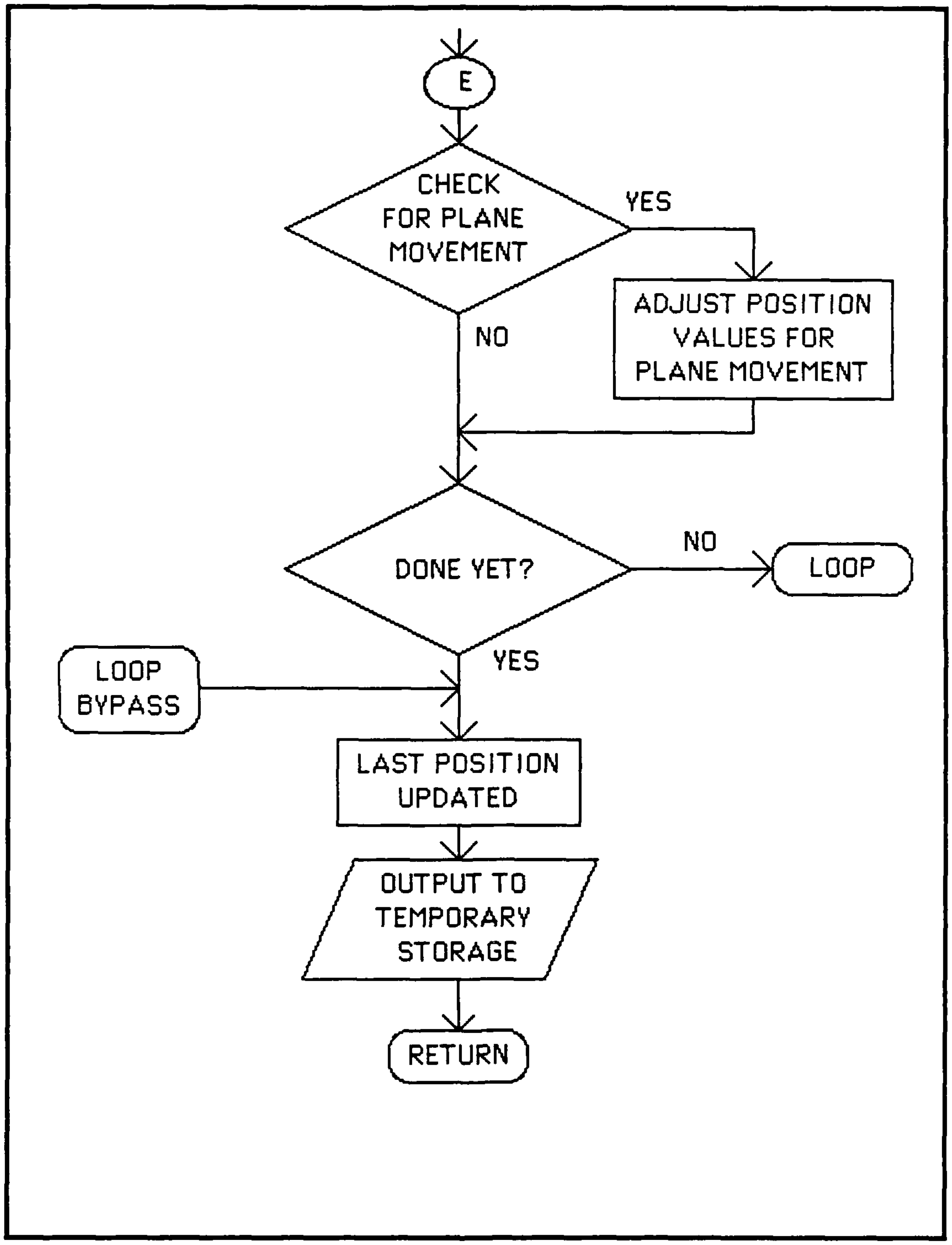


Figure 5.7 (Cont) Subroutine FHOME. Finish to Home.

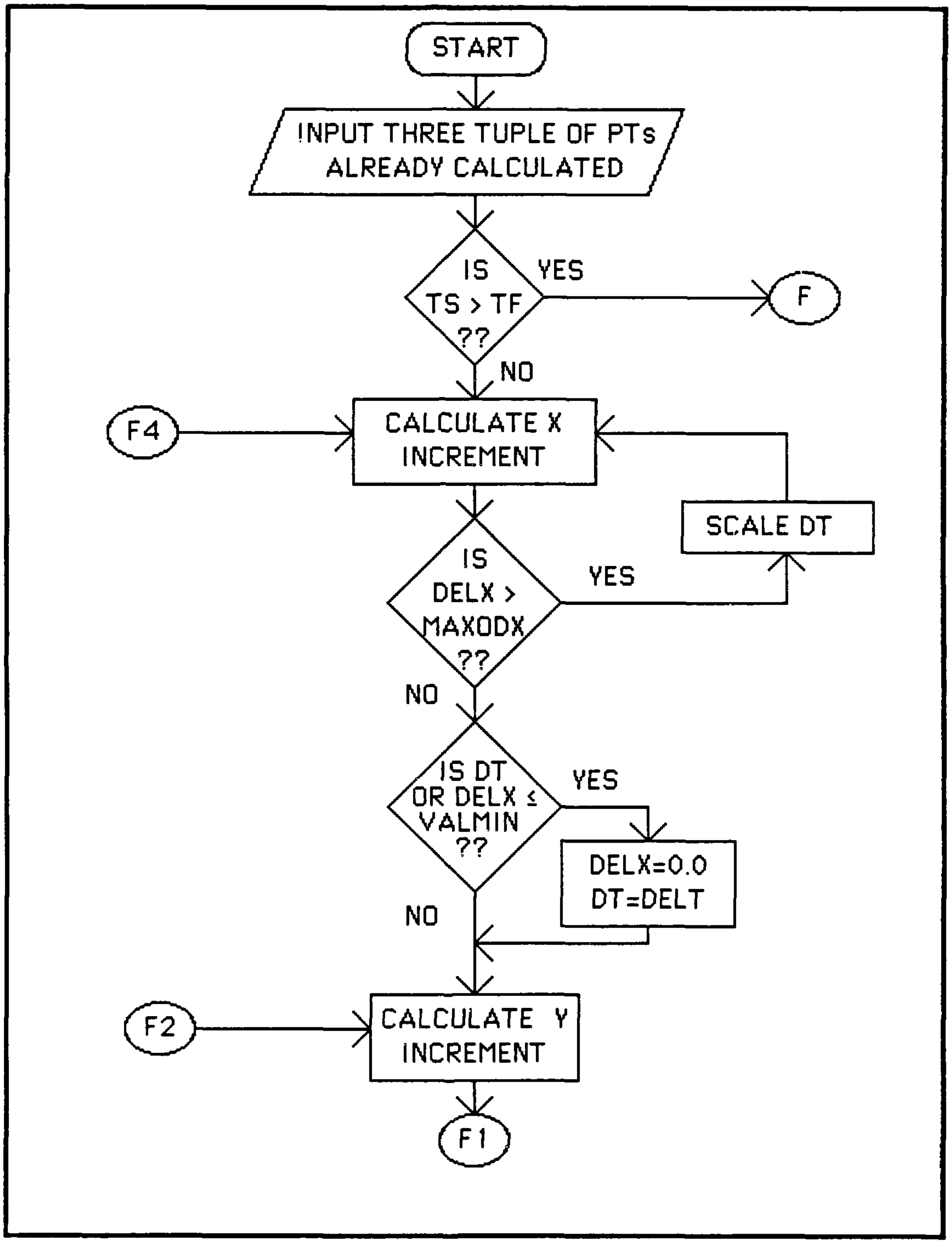


Figure 5.8 Subroutine SFPAR . Parametric Start to Finish.

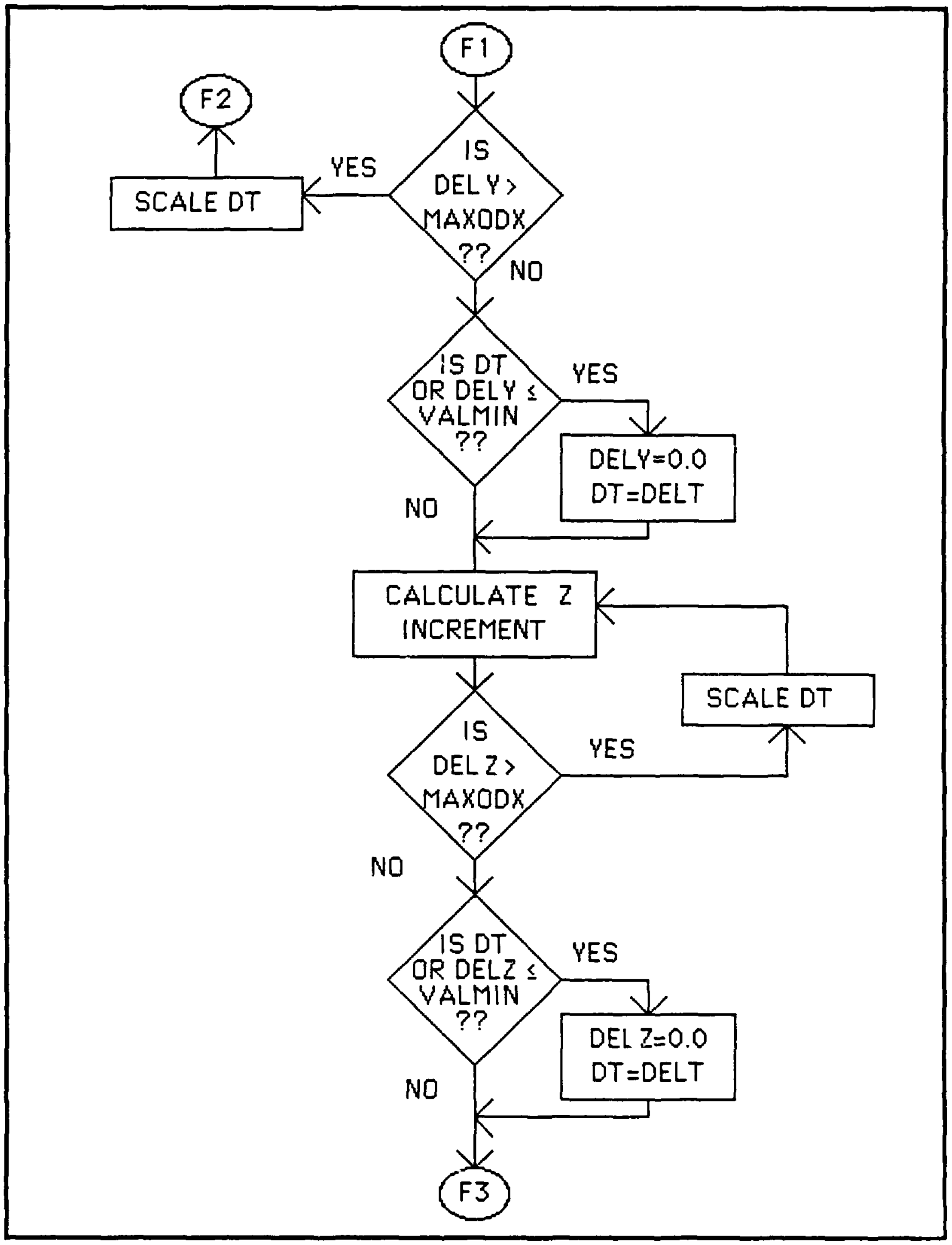
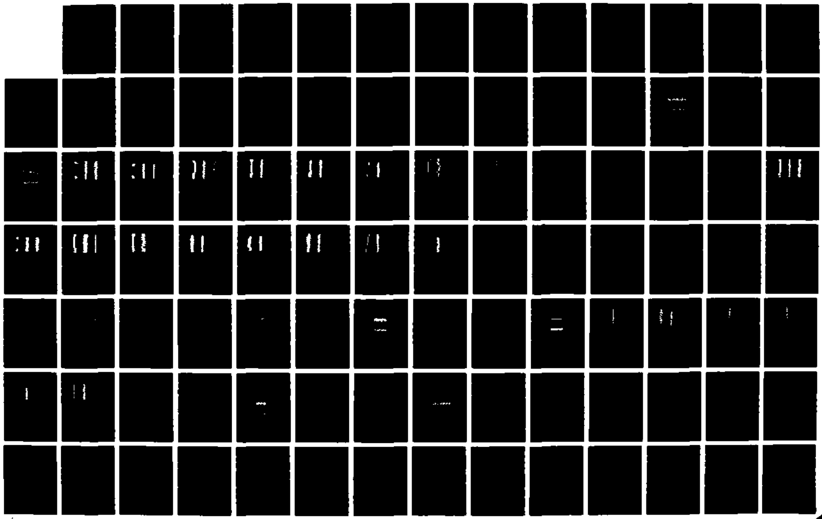
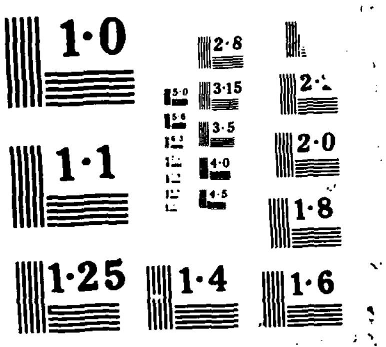


Figure 5.8 (Cont) Subroutine SFPAR. Parametric Start to Finish

AD-A190 960 PATH FOLLOWING ROBOT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S G GOODWAY DEC 87 2/4

UNCLASSIFIED F/G 12/9 NL




1·0
2·8
3·15
3·5
4·0
4·5
2·0
1·1
1·8
1·25
1·4
1·6

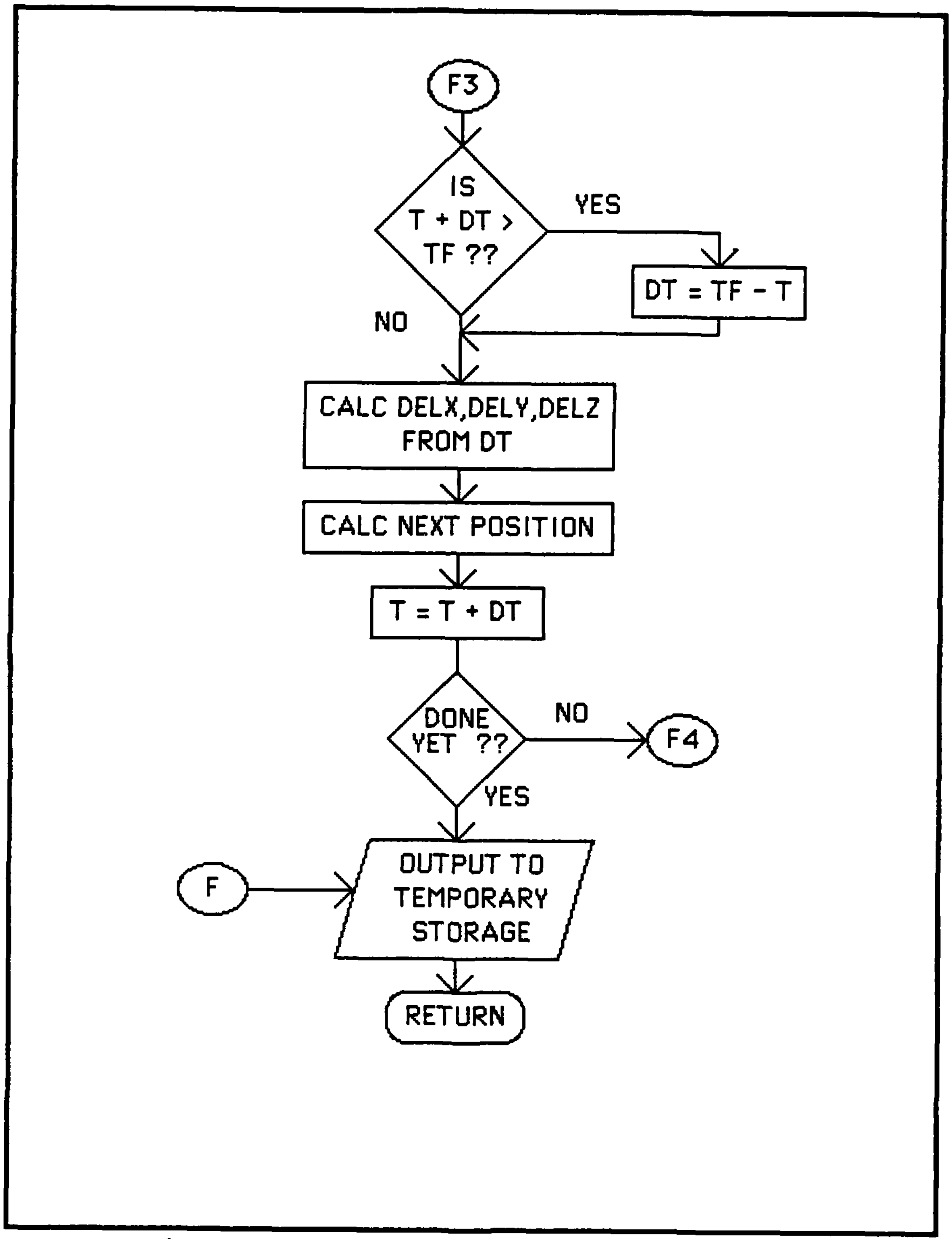


Figure 5.8 (Cont) Subroutine SFPAR. Parametric Start to Finish.

VI. PATH FOLLOWING ROBOT SIMULATIONS

A. INTRODUCTION

Interfacing the path program developed in Chapter Five with the DSL program selected in Chapter Four has resulted in some interesting observations. Since the overall objective of the thesis is to get a cartesian robot arm to follow a path that was precalculated it would seem that all that needs to be done is to generate a set of coordinates and sequentially feed these values into the robot arm. However the DSL program, which is chosen to simulate the robot arm has certain control parameters that work within the bounds of DSL and do not give an indication of the type of hardware necessary to accomplish the task. For example, DSL has several options for the way it accepts inputted data and applies this data to the robot arm. Two of these were discussed in Chapter Four but never explained. If the robot were to receive a set of coordinates it is fairly easy to see how to make the STEP input, but not readily apparent on how to take the set of coordinates and get a RAMP. Subdividing the coordinates in a linear fashion and inputting these at a constant rate is one answer, but that is the same as defining smaller increments in the path data and is less accurate.

Another problem occurs when using position as a means of deciding whether or not you have reached the end of the step and are ready to input the next set of data. Using position can be accurate at the points that were desired in the path equation but you do not really have sufficient control over the path. It is probably better to set a designated time interval for each path length and use this to control the rate at which the robot arm gets the next set of coordinate points. This was also discussed in Chapter Four.

B. SCALING

The path generating program also has made no mention of the length of the robot arms. Each robot arm can be of different length and blindly assuming that the robot can follow all of the points without any problem may lead to serious problems. The DSL program also assumes that the lengths of the arms are sufficient. Some prescaling may be needed to make all of the movements within the range of the arm. This would take the maximum value of the coordinate listing and scale all of the coordinate points with the relative length of the arm in that direction. This would lead to a distortion of the equations used to generate the input. An alternative method would be to scale everything with reference from the largest value to a unit length, causing no distortion, and then setting all of the values on the smaller robot arms to a max value while that position is being attempted. It would be similar to running the arm up against a stop point and maintaining that value for all of the points that occur outside of that region.

It is the latter method that will be chosen in this thesis. After the Path generation program has been completed a path boundary program will take the maximum value of the input coordinate points and scale the array so that this value is one. For now we will assume that all arms are of equal length and have a value of one. Of course it may be that a user would want a larger scale value and with a minor modification that can be entered.

C. TEST RUNS

There are several types of paths that need to be considered and tested to insure proper operation. They are:

1. Linear
2. Large curvature
3. Small curvature
4. Repeated motions

The curvature paths can be observed by using conic sections, which include mainly circles, ellipses, parabolas, and hyperbolas, or transcendental functions, which are mainly composed of sine, cosine, or tangential functions. Since it would be difficult to test all of the different types of curves we have arbitrarily chosen a circular function and a sine function as test platforms. For a repeated motion the right circular helix has been chosen. These movements will also insure proper operation as a curve moves tangent to a coordinate plane. The final motions will be that of a straight line in all of the coordinate axes and one parallel to each coordinate axis. The goal is to demonstrate flexibility of the program and find problem areas.

Table 6.1 summarize the test runs that that will be studied. Each of these runs will be applied to both the STEP and RAMP excitations and Position and Velocity Methods discussed in Chapter 4. Figure 6.1 summarizes the equations used and the modifications to the PATH program.

TABLE 6.1
TEST RUNS

TEST RUN	EXCITATION	METHOD	MOTION
1(a)	STEP	POSITION	Linear motion
1(b)	STEP	VELOCITY	
1(c)	RAMP	POSITION	
1(d)	RAMP	VELOCITY	
2(a)	STEP	POSITION	Circular motion
2(b)	STEP	VELOCITY	
2(c)	RAMP	POSITION	
2(d)	RAMP	VELOCITY	
3(a)	STEP	POSITION	Helical Motion
3(b)	STEP	VELOCITY	
3(c)	RAMP	POSITION	
3(d)	RAMP	VELOCITY	
4(a)	STEP	POSITION	Sinusoidal
4(b)	STEP	VELOCITY	
4(c)	RAMP	POSITION	
4(d)	RAMP	VELOCITY	

D. RESULTS

1. Linear Motion

Linear motion is one of the more basic movements and also one of the hardest to describe. For example a linear movement parallel to any axis is hard to put to an equation while in the cartesian coordinate system if not using parametric equations. Two linear motions were chosen to show that the program works correctly in both the cartesian and the parametric equation form. They are described by:

Linear path 1:			
Equation 1 :	(1.0,0.25,0.25)	to	(0.25,1.0,1.0)
Equation 2 :	(0.5,1.0,1.0)	to	(1.0,0.5,1.0)
Linear path 2:			
Equation 1 :	(1.0,0.25,0.5)	to	(1.0,0.75,.05)
Equation 2 :	(0.75,1.0,.05)	to	(0.25,1.0,0.5)
Equation 3 :	(0.0,1.0,0.0)	to	(0.0,1.0,1.0)
Equation 4 :	(0.75,0.25,0.25)	to	(0.25,.075,0.75)

Linear Path 1 was calculated on an early path generating program using only cartesian coordinates while Linear Path 2 was generated on the final path program mixing cartesian and parametric equations. Linear Path 1 was re-run on the final path program to insure proper operation. Parametric equations were used to generate the paths parallel to the axis while cartesian equations were used to generate diagonal paths (however parametric equations would be just as easy or easier to use). Equations used to describe Linear Path 1 and 2 are outlined in Figure 6.1. Results of the Linear Path 1 and 2 are displayed in Figures 6.2 (a)-(d) and 6.3 (a)-(d), respectively, and show a 3-dimensional view of the desired path, the Robot arm position with respect to time for each degree of motion, each of the axis plane projections, and the error between the desired path inputted and the final arm position resulting from the simulations for the STEP excitation, Position method of the conditions shown in Table 6.1 (1a). The other conditions listed in Table 6.1 for linear motion were run but showed nothing significant that will not be discussed in more detail in the next section on circular motion and are located in Appendix H. Observing the time versus distance curves shows smooth motion for the size of step inputted while the plane projection plots show that the path appears to be following the equations accurately. Path error generated in each of the coordinate axes show that each of the arm axes are within the desired 0.001 inches of accuracy [VALMIN in the simulation program]. The error developed by the path, model, and simulation will be discussed later in this chapter.

2. Circular Motion

Test runs 2(a)-(d) studied the effects of circular motion. First a single quadrant (ROBPATH3.FORTRAN) was run to see the effects on the curvature. Then when attempting to do a right circular helix (ROBPATH5.FORTRAN) a problem was noted when the x increment was approaching a tangent with the y coordinate axis (at x=0). That was when the decision to do a complete circle was made (ROBPATH4.FPRTRAN). Being more interesting than one quadrant most of the testing will be done on the complete circular motion.

The circular path generated was linear from home to start of the circular motion using the HSTAR subroutine with a maximum step size of 0.1 units. Then the path followed the equations:

Equation 1:

$$y = ((.25 - (x-.5)^2)^{1/2} + .5) \qquad \text{[eqn 6.1]}$$

$$z = 0.5 \qquad \text{[eqn 6.2]}$$

Equation 2:

$$y = (-(.25 - (x-.5)^2)^{1/2} + .5) \qquad \text{[eqn 6.3]}$$

$$z = 0.5 \qquad \text{[eqn 6.4]}$$

which describe circular motion in the XY plane, with an increment maximum step size of 0.001. The change in maximum step size was implemented to show that the path can be made smooth. This technique will be implemented only for the position method techniques because implementing this technique for the velocity method would require significant changes in the DSL simulation and would affect the outcome of the simulation. For the velocity method a maximum step size of 0.01 units was used for all of the paths. After finishing the path movement a linear path to home was completed with a maximum step size of 0.1 units.

Figure 6.4 (a) shows time versus simulated arm position for each of the coordinate axes while Figure 6.4 (b) shows the plane projections for each of the coordinate axes. To do this motion a total of 505 coordinate points were computed and inputted to the simulation. As each step reached the desired accuracy, stipulated by VALMIN of the simulation program [Position Method], the next three tuple of position points were inputted and the simulation was run again. The curve is very smooth and quite accurate. Figure 6.4 (c) plots the error between each of the desired inputted points and the resulting simulation movement. As you can see the accuracy at the final positions are well within the accuracy described by the desired Position Method value of 0.001 inches.

Figures 6.5 (a)-(c) represent the circle path using the RAMP excitation along with the Position Method of simulation control. The time response curves show a more linear response especially near discontinuities where radical changes in slope occur. For example when comparing Figures 6.4 (a) and 6.5 (a), near the peaks and valleys of the time response curves, there is a definite rounding to the STEP Excitation time response curves.

Also notice that there is a significant reduction in time when using the RAMP Excitation over the STEP Excitation. Path error is nearly the same for both the STEP Excitation, Figure 6.4 (c), and the RAMP Excitation, Figure 6.5 (c), however it is interesting to note that the RAMP Excitation error decreases as a function of time. There is also a time delay at the start of the time response curves which is due to the first step of the RAMP being zero and the desired path being zero at the same time. This phenomenon will be observed in some of the latter RAMP excitation curves also.

Figures 6.6 (a)-(l) and Figures 6.7 (a)-(k), for the STEP and RAMP excitation, respectively, show the effects of inputting the "next" position at shorter intervals of time. This method is called the Velocity Method. Due to the large number of figures required only the XY plane projection and the path errors for each axis will be shown. The test runs for the Velocity and Position methods are listed in Table 6.2. Comparison of the XY plane projections for the STEP Excitation, Velocity Method show that the error in the paths stay to within 0.001 inches until the inputted time interval reaches 0.008 seconds when there is a sharp increase in the path error in the X direction. The Y and Z path error remain constant until 0.005 seconds when they start rising at about the same rate as the X error. When RAMP Excitation was applied to the simulation program with Velocity Method, similar results were experienced except that the error curves exceeded 0.001 inches with a 0.001 secs shorter input interval in all three axis. Table 6.2 contains an estimation of the maximum error for both the STEP and RAMP Excitation and the results are plotted in Figures 6.8 (a)-(c). Figure 6.6(l) was used to verify the DSL Program save function to insure that the same plot was produced when the save statement was larger than the velocity step increment. For example, whether the save statement for a VAIMIN of .001 is saved at time intervals of .005 or .001, we arrive at the same result.

TABLE 6.2

CIRCULAR MOTION TEST RUNS/ VELOCITY METHOD

POSITION METHOD:

PATH PROGRAM	: ROBPATH4.FORTRAN
SIMULATION PROGRAM	: ROBSIM2T.FORTRAN or RSIM2T6.FORTRAN
DATA FILE	: PATH4.DATA

FIGURE NO.	EXCITATION
6.4 (a)-(d)	STEP
6.5 (a)-(c)	RAMP

VELOCITY METHOD:

PATH PROGRAM	: ROBPATH4.FORTRAN
SIMULATION PROGRAM	: ROBSIM2V.FORTRAN or RSIM2V6.FORTRAN
DATA FILE	: PATH4.DATA

PEAK ERROR [milli-inches]

INPUT		STEP EXCITATION (FIG 6.6)			RAMP EXCITATION (FIG 6.7)		
INTERVAL/(FIG)		X	Y	Z	X	Y	Z
0.020	(a)	0.10	0.60	0.15	0.10	0.50	0.15
0.015	(b)	0.10	0.60	0.18	0.10	0.50	0.20
0.010	(c)	0.25	0.60	0.18	0.08	0.50	0.18
0.008	(d)	1.40	0.60	0.18	0.25	0.50	0.18
0.007	(e)	2.50	0.60	0.18	0.90	0.50	0.19
0.006	(f)	4.90	0.50	0.25	2.50	0.50	0.19
0.005	(g)	8.00	1.00	0.70	5.00	0.50	0.19
0.004	(h)	15.0	2.50	1.80	10.0	0.60	0.30
0.003	(i)	30.0	6.50	4.50	22.0	0.60	1.40
0.002	(j)	80.0	20.0	12.0	50.0	6.00	5.00
0.001	(k)	300.0	85.0	60.0	200.0	35.0	30.0
0.001*	(l)	300.0	85.0	60.0			

Differences in the Position and Velocity method can be seen by comparing STEP Excitation, Position Method with STEP Excitation, Velocity Method, where the desired position is inputted at 0.008 seconds so that the total time is around 4.0 seconds. The X path error is larger in the Velocity Method while the Y and Z path error is less. The plane projections appear identical, however the time response are dissimilar in both the X and Y direction. The X direction is more rounded through the circular part for the Velocity Method [see Appendix H for the Velocity Method Time Response Figures, VALMIN= 0.008].

3. Helical Motion

With the current path formation program it is difficult to do repetitive movements, unless all of the dimensions are repetitive in nature. The circle discussed above was repetitive in all 3 dimensions. When an attempt is made to form even the simplest right circular helix it is necessary to keep track of the number of revolutions and the process becomes more difficult. To solve this problem a parametric subroutine was added to the path generating program that allows X, Y, and Z to be expressed as a function of a single variable, such as time or displacement. This reduces the number of equations needed to do a right circular helix to three. The parametric equations and modifications to ROBPATH5 program are listed in Figure 6-9. Two runs were made for a three and six revolution right circular helix and are show in Figures 6.10 (a)-(d) and 6.12 (a)-(d) for STEP Excitation and Position Method, and Figures 6.11 (a)-(c) and 6.13 (a)-(d) for RAMP Excitation, Position method. Figure 6.14 and 6.15 (a)-(c) show different STEP Excitation, Velocity Method Simulations. To arrive at the same accuracy as the Position Method it is necessary to have Velocity Method Input Intervals (VMII) of 0.020 seconds which is about the same as

(number of points) x (VMII) = Position completion time

which is as expected. Figure 6.18 shows the VMII versus path errors overlayed onto the circular path Figures 6.8 (a) and (b).

To insure proper operation of the parametric equation the start point for the independent variable must be smaller than the finish point. In these examples the independent variable t was chosen to be in radians and the equations adjusted as needed to get the desired output. Figure 5-8 shows the flowchart of the subroutine used. The three revolution right circular helix took 231 input position points to simulate while the six revolution right circular helix took 420 input positions. Figures 6.17 (a) and (b) confirm that the VMII for the six revolution right circular helix with ramp excitation would be between 0.020 and 0.015 seconds. Calculating directly indicates that a value of 0.0155 would be adequate to drive the Velocity Method path error under 0.001 inches.

4. Sinusoidal Motion

Two sinusoidal equations that were generated are:

Y = f(X) = ABS (SIN (X)) [eqn 6.5]

Z = f(X,Y) = ABS (COS (2*PI*X)) [eqn 6.6]

The STEP excitation, Position method results are shown in Figure 6.19

(a)-(d) along with a three dimensional diagram of the motion, while the RAMP Excitation, Position Method results are shown in Figure 6.20 (a)-(c). The plots are correct but are not of much use. There are a couple of points of discontinuity which are caused by interaction between the large distances between steps inputted and the absolute function.

E POSITION ACCURACY

When comparing STEP or RAMP Excitations, with Position Method simulation and varying the value of VALMIN, which is the error difference between the desired position (RX,RY, RZ) and the motor position (C1, C2, C3), while the Path Error is the difference between the desired position (RX,RY,RZ) and the simulated position (CX, CY, CZ). There is a limit to how accurate the simulation can get by just changing VALMIN. Table 6.3 list the error for each of the axis when following the circular path and the three revolution right circular helix using the Position Method. Only 50 position steps were utilized to keep computing time reasonable.

Since there was no movement in the Z direction for the circular path there is no Path Error in that direction. This was the reason that the helical path was run also. In most cases the path error is half of the desired VALMIN. The only disagreement is in the Y direction where there seems to be a limit where the Y path error levels off regardless of the value of VALMIN. This is around 0.080 milli-inches for the Circular Path and 0.060 milli-inches for the Helical Path. Smaller values for the circular path and the helical path indicate a lower limit in the simulation program to arrive at a desired accuracy. XY Plane Projection and X, Y, and Z Path Errors are for the Velocity Method are located in Appendix H.

TABLE 6.3
ACCURACY USING POSITION METHOD

VALMIN	CIRCULAR PATH ERROR [milli-in]				HELIX PATH ERROR [milli-in]			
[milli-in]	X	Y	Z	FIG.	X	Y	Z	FIG.
50.0	---	---	---	---	25.0	25.0	1.5	H-4(a)
10.0	5.0	5.0	0.0	H-3(a)	5.0	5.0	0.1	H-4(b)
5.0	2.5	2.5	0.0	H-3(b)	2.5	2.5	0.12	H-4(c)
1.0	0.5	0.5	0.0	H-3(c)	0.5	0.5	0.16	H-4(d)
0.5	0.25	.25	0.0	H-3(d)	0.25	0.3	0.11	H-4(e)
0.1	0.05	0.08	0.0	H-3(e)	0.05	0.06	0.05	H-4(f)
0.05	0.025	0.08	0.0	H-3(f)	0.03	0.06	0.03	H-4(g)
0.01	0.018	0.09	0.0	H-3(g)	-----	FAILED	-----	
0.005	0.02	0.08	0.0	H-3(h)				
0.001	-----	FAILED	-----					

Failed runs were due to large computation times, or problems with the simulation in resolving integrations. Several different integration techniques were attempted, along with various DELT and DELS values, but the runs did not improve or complete more than a few path points out of the fifty in the path, while the paths that were completed successfully were rapidly done.

F. ERROR

There are several areas of error involved when trying to get accurate movement of the robot arm. They include:

1. Motor simulation model error in assuming that the model can be reduced to a second order system.
2. Robot Kinematics and Dynamic equation assumptions.
3. Computer roundoff errors. While DSL runs in double precision there are still minimum accuracies that are fairly large when compared to the accuracies that are needed in the simulation. All programs other than DSL are single precision and susceptible to large truncation and round-off errors.
4. Program design errors. The largest error involved in the program is in assuming a minimum value for several of the operations. In the Position Method the value VALMIN (E) is chosen to be a given value. The difference between C1 and CX, C2 and CY, and C3 and CZ is another error. Ideally these values will be the same, but that is not always the case. In the previous section the fact that the error in the Y direction can be larger than the value of VALMIN shows that CY is not the same as C2 by a considerable amount. This shows a discrepancy in the computer simulation model and in the way that Path Error was defined.

The effect of these errors is to create a sphere of error around each of the position points. Not necessarily the same in all three coordinate axes, it is easier to visualize a sphere following the desired path. As the Robot arm follows the desired path around, the error effectively becomes a cylinder around the desired path. Added to the non-linearity of the Robot Arm between points the error cylinder can become quite large.

Path Equations for Linear Path 1

EQUAT1 = 0.500
EQAT1Y = 1.25 - X

EQUAT2 = 1.00
EQAT2Y = 1.50 - X

Start and Finish Points For Linear Path 2

XS1	=	1.00	XS2	=	0.50
XF1	=	0.25	XF2	=	1.00

Path Equations for Linear Path 2

PEQX1 = 0.75
PEQY1 = 0.5 + 0.25 T
PEQZ1 = 0.00

PEQX2 = 0.75 - 0.5 T
PEQY2 = 0.75
PEQZ2 = 0.25

PEQX3 = 0.25
PEQY3 = 0.75 -0.5 T
PEQZ3 = 0.00

PEQX4 = 0.25
PEQY4 = 0.25 + 0.75 T
PEQZ4 = 1.00

EQUAT5 = X
EQAT5Y = 1.5 - X

Start and Finish Points For Linear Path 2

TS	=	0.00	XS5	=	0.50
TF	=	1.00	XF5	=	1.00

Figure 6.1 Path Equations and Program Modifications for Linear Path 1 and 2.

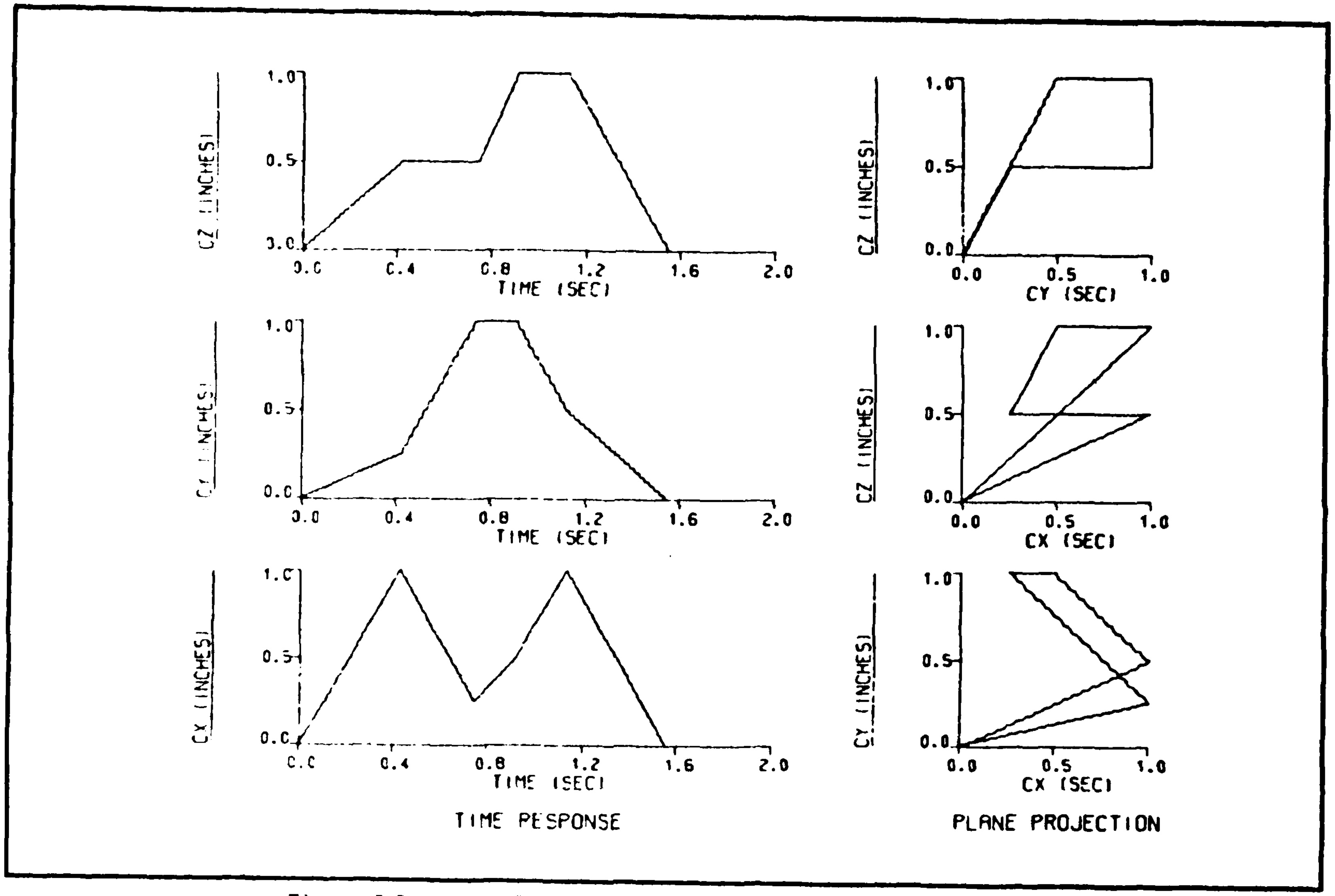


Figure 6.2 Linear Path 1, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

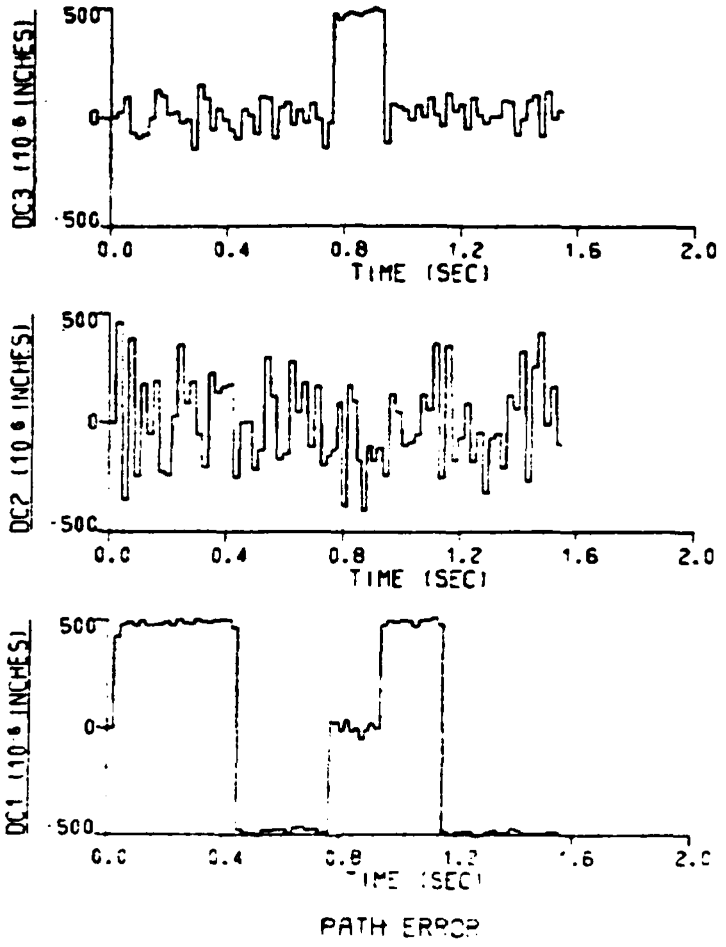


Figure 6.2(c) Linear Path 1, STEP Excitation, Position Method. X, Y, and Z Path Error.

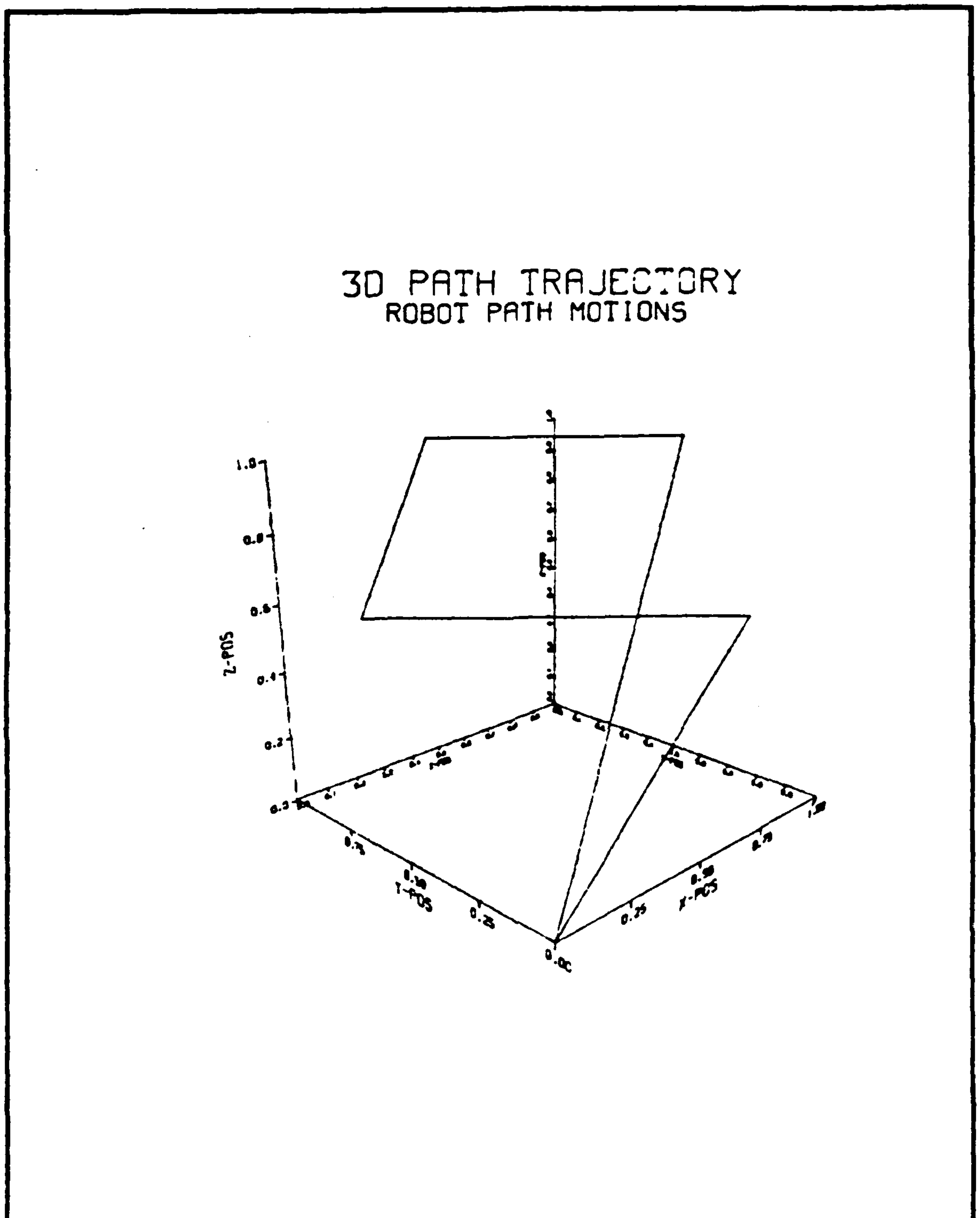


Figure 6.2(d) 3-Dimensional View of Linear Path 1.

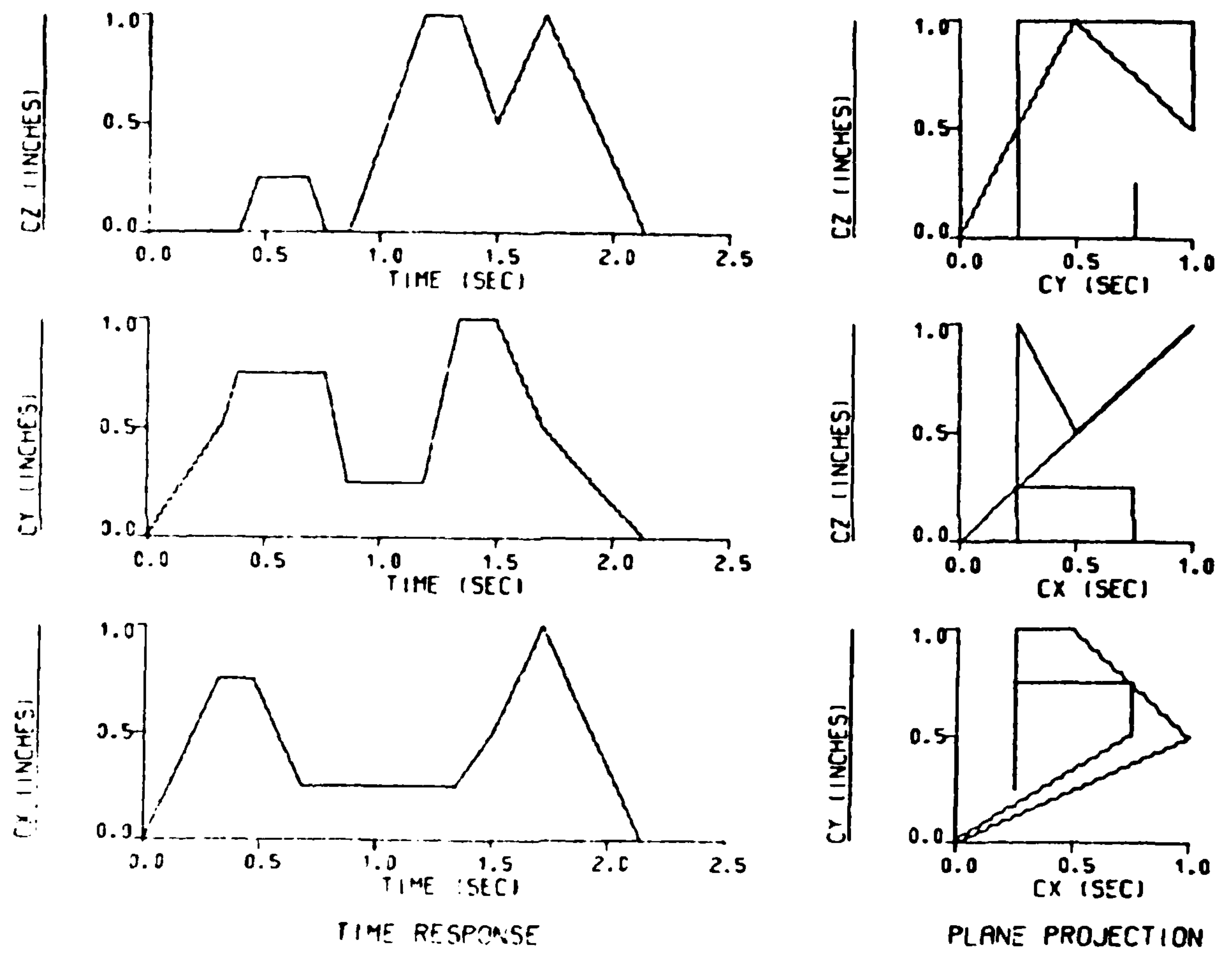


Figure 6.3 Linear Path 2, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

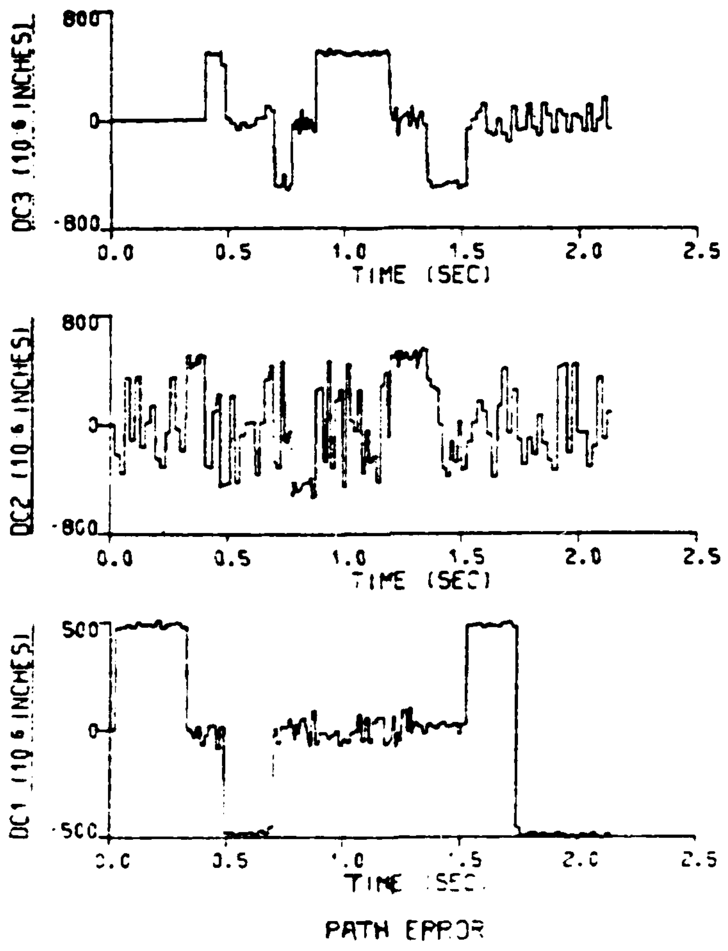


Figure 6.3(c) Linear Path 2, STEP Excitation, Position Method. X, Y, and Z Path Error.

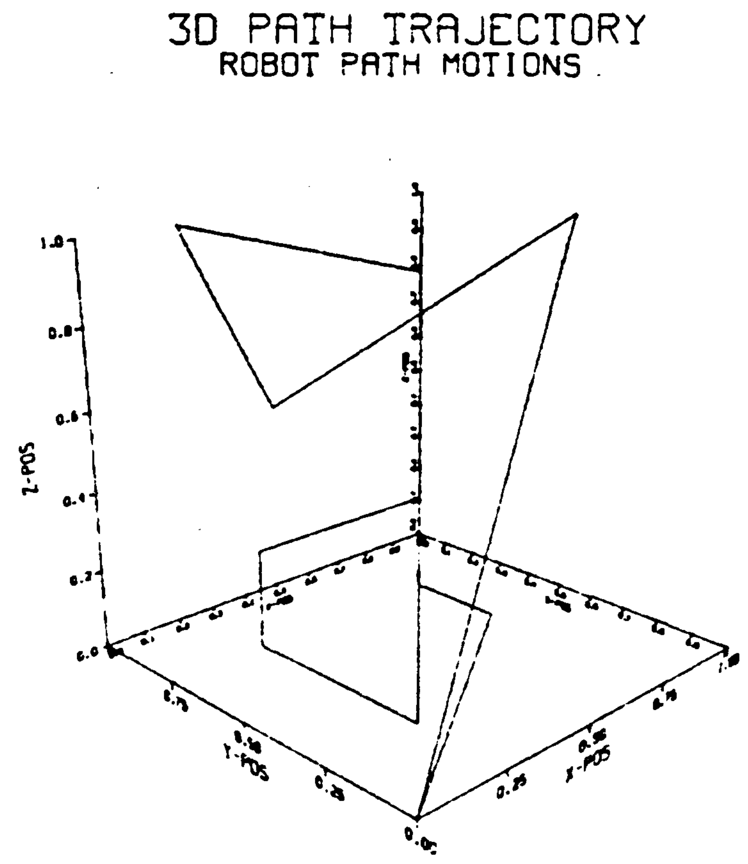


Figure 6.3(d) 3-Dimensional View of Linear Path 2.

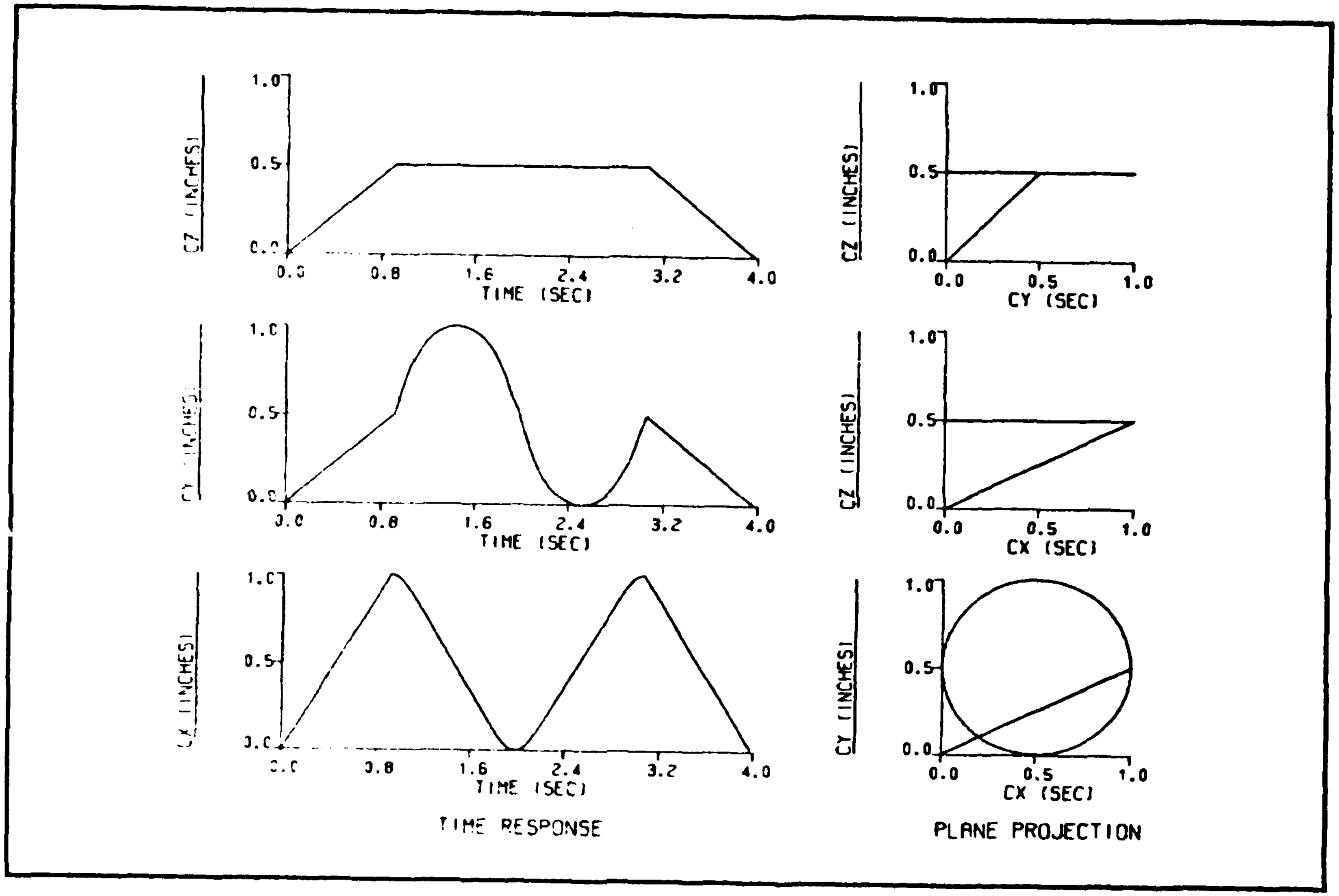


Figure 6.4 Circular Path, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

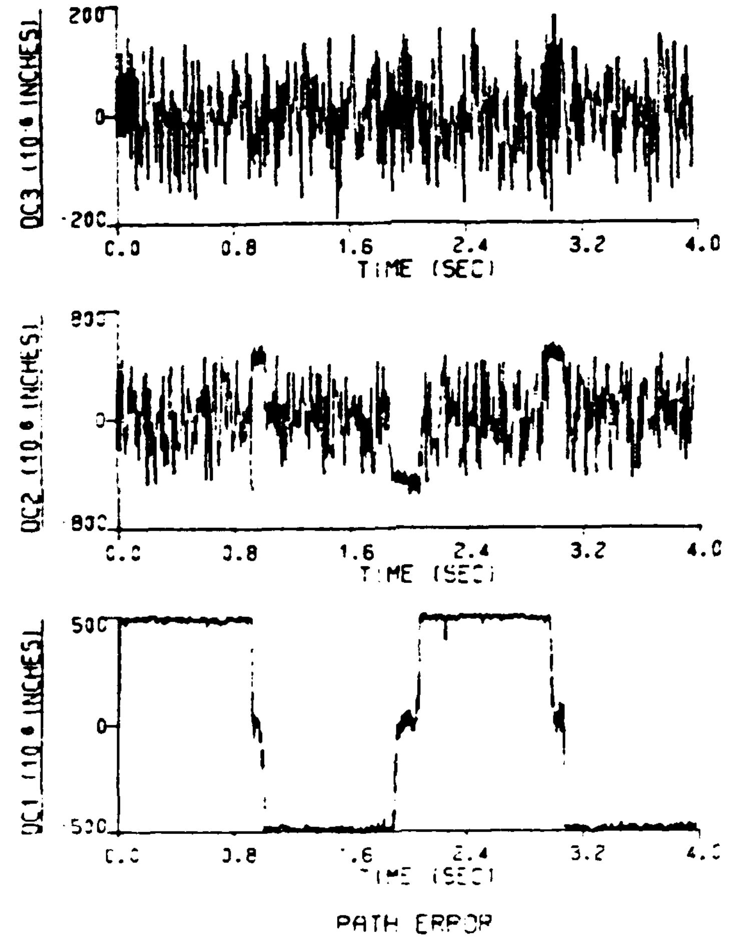


Figure 6.4(c) Circular Path, STEP Excitation, Position Method. X, Y, and Z Path Error.

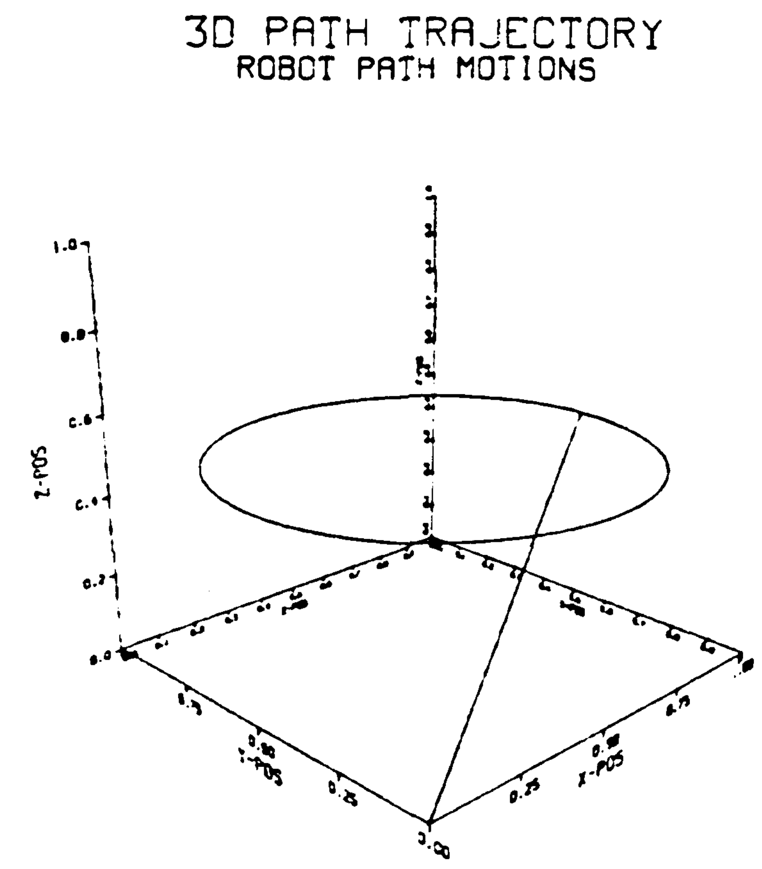


Figure 6.4(d) 3-Dimensional View of Circular Path.

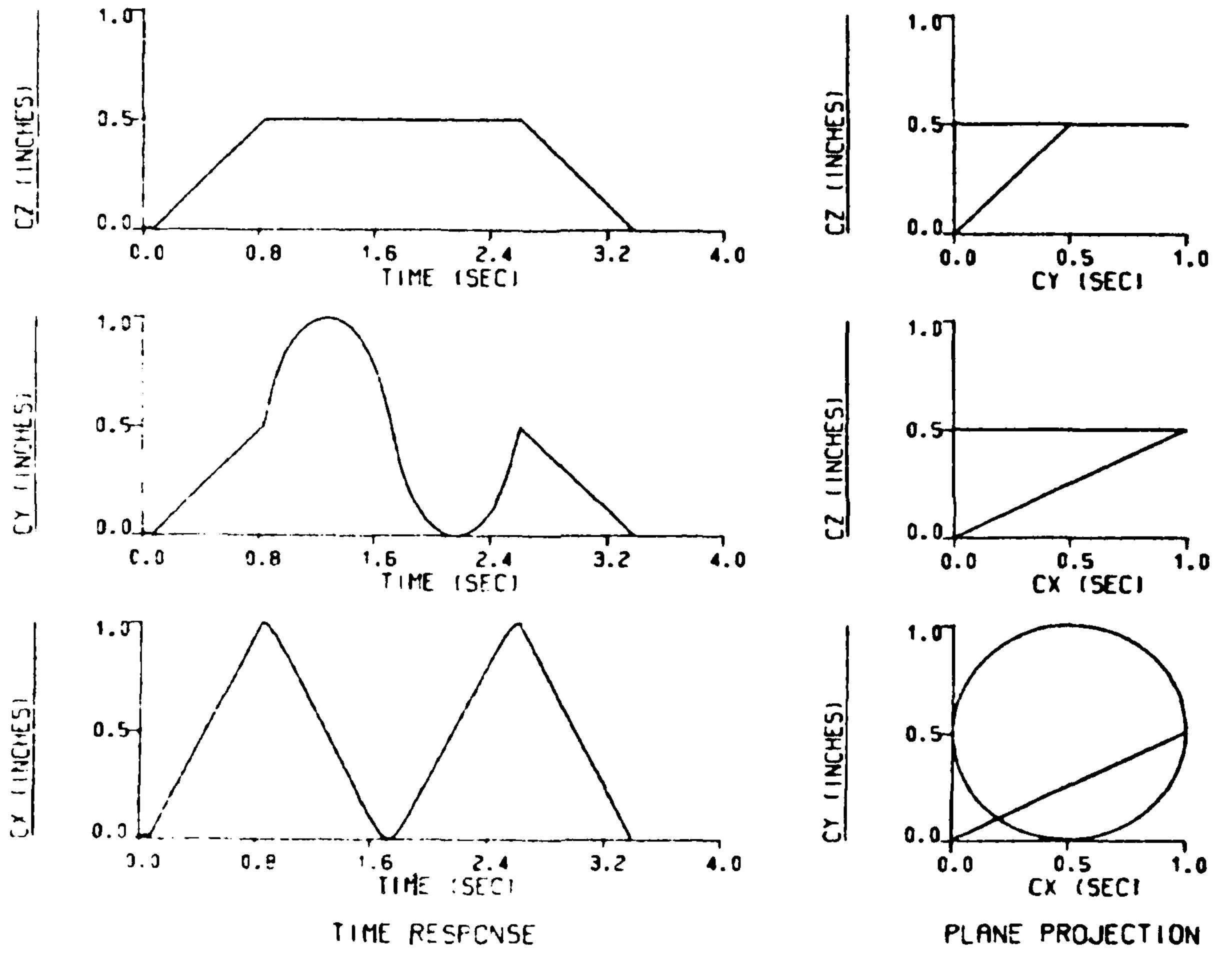


Figure 6.5 Circular Path, RAMP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

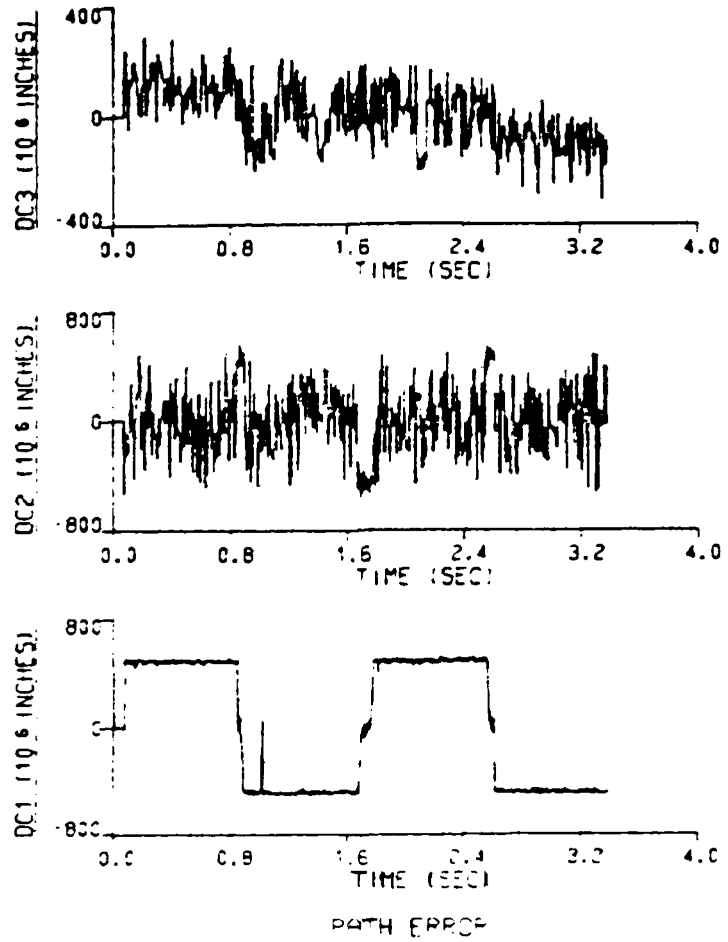


Figure 6.5(c) Circular Path, RAMP Excitation, Position Method.
X, Y, and Z Path Error

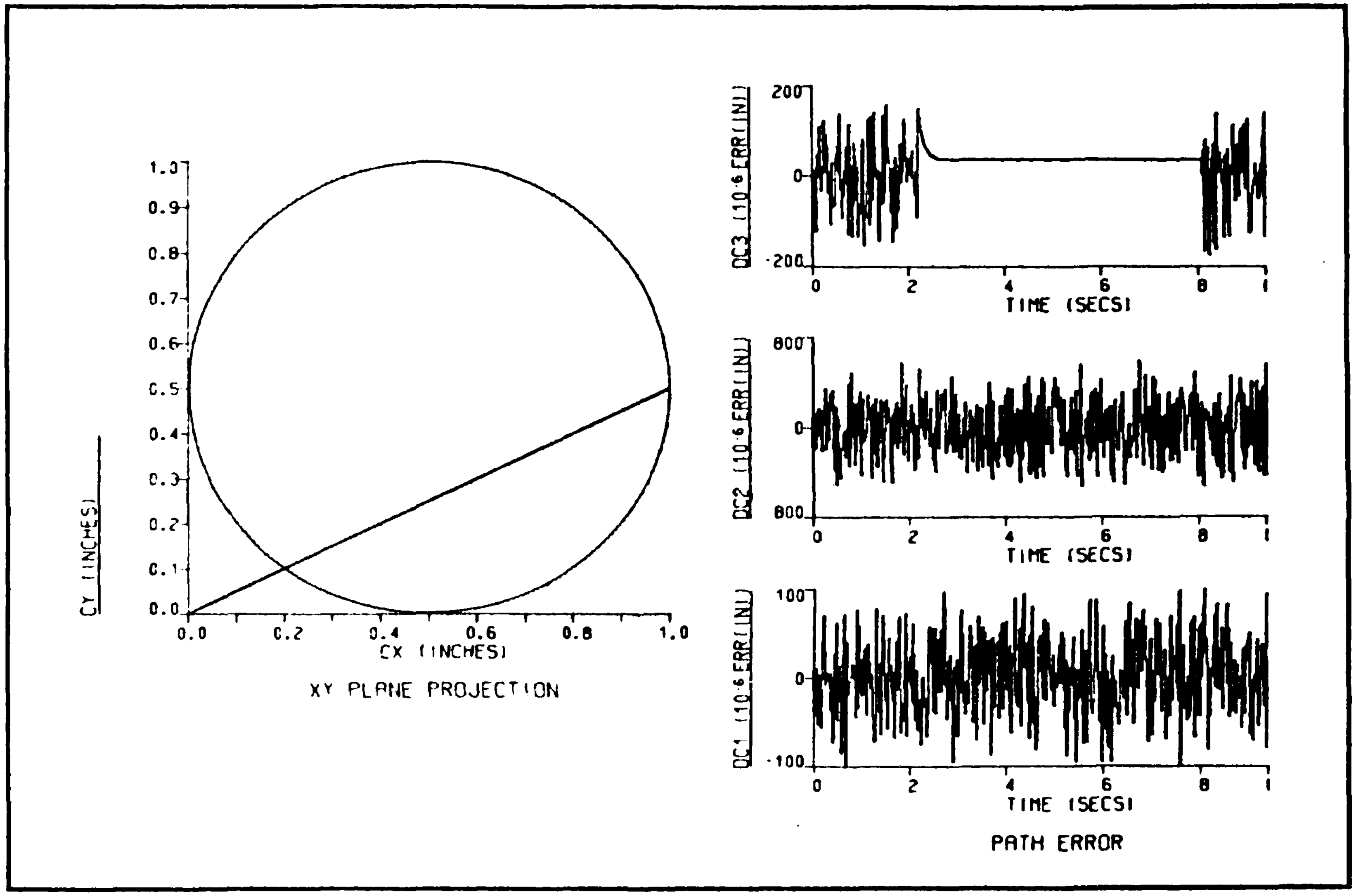


Figure 6.6(a) Circular Path, STEP Excitation, Velocity Method. [T=0.020]. XY Projection and X, Y, and Z Path Error.

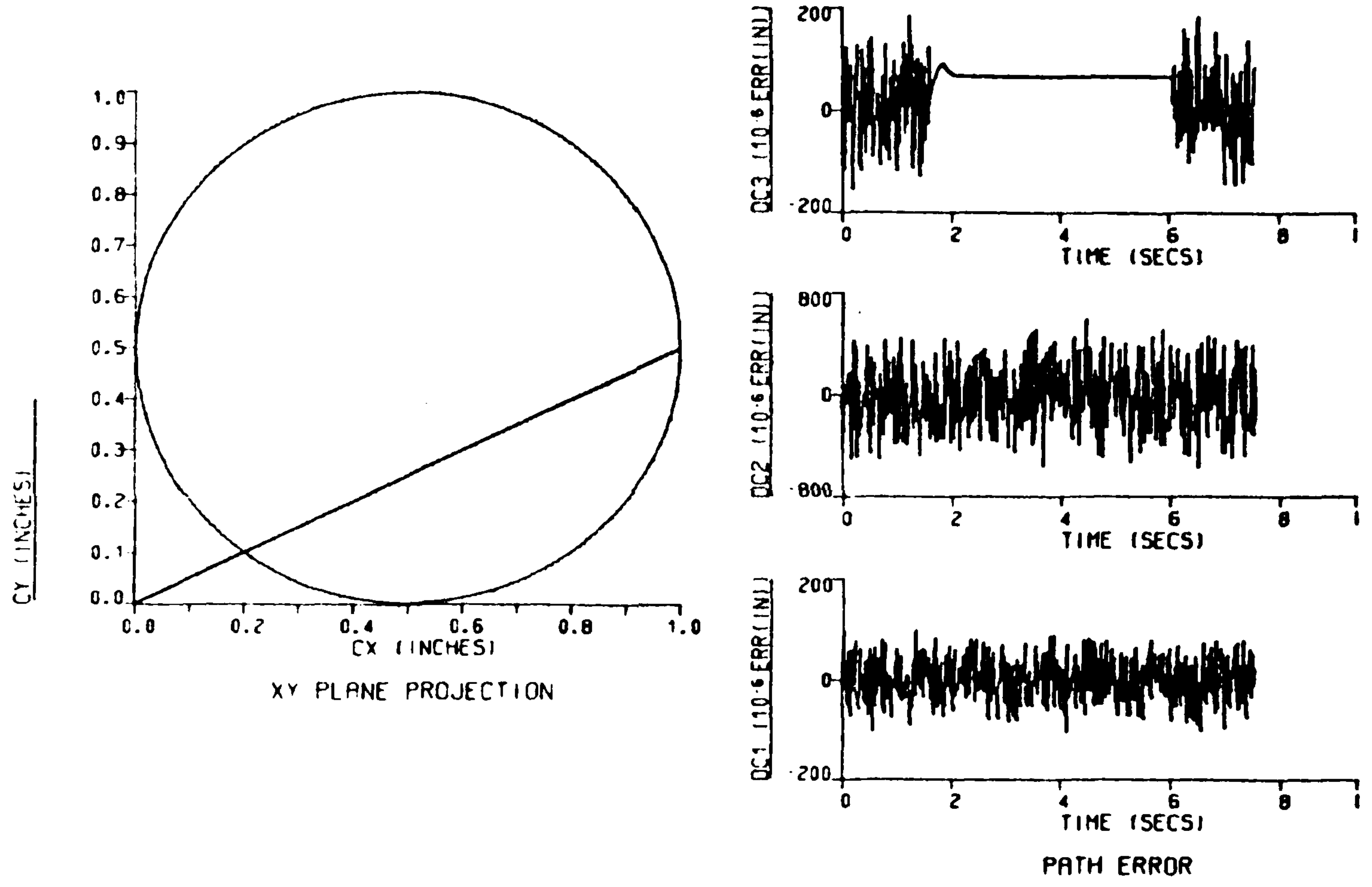


Figure 6.6(b) Circular Path, STEP Excitation, Velocity Method. [T=0.015]. XY Projection and X, Y, and Z Path Error.

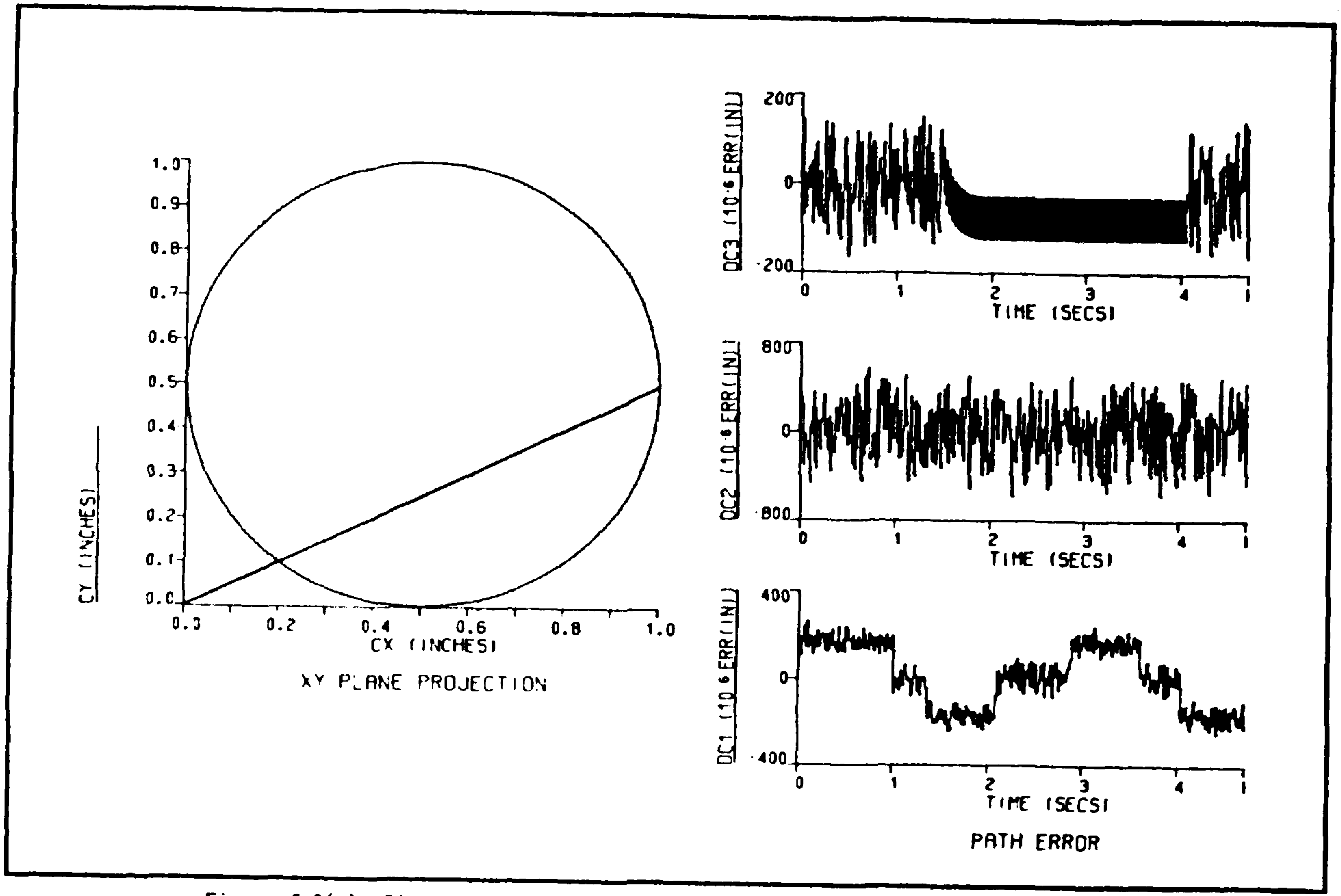


Figure 6.6(c) Circular Path, STEP Excitation, Velocity Method. [T=0.010]. XY Projection and X, Y, and Z Path Error.

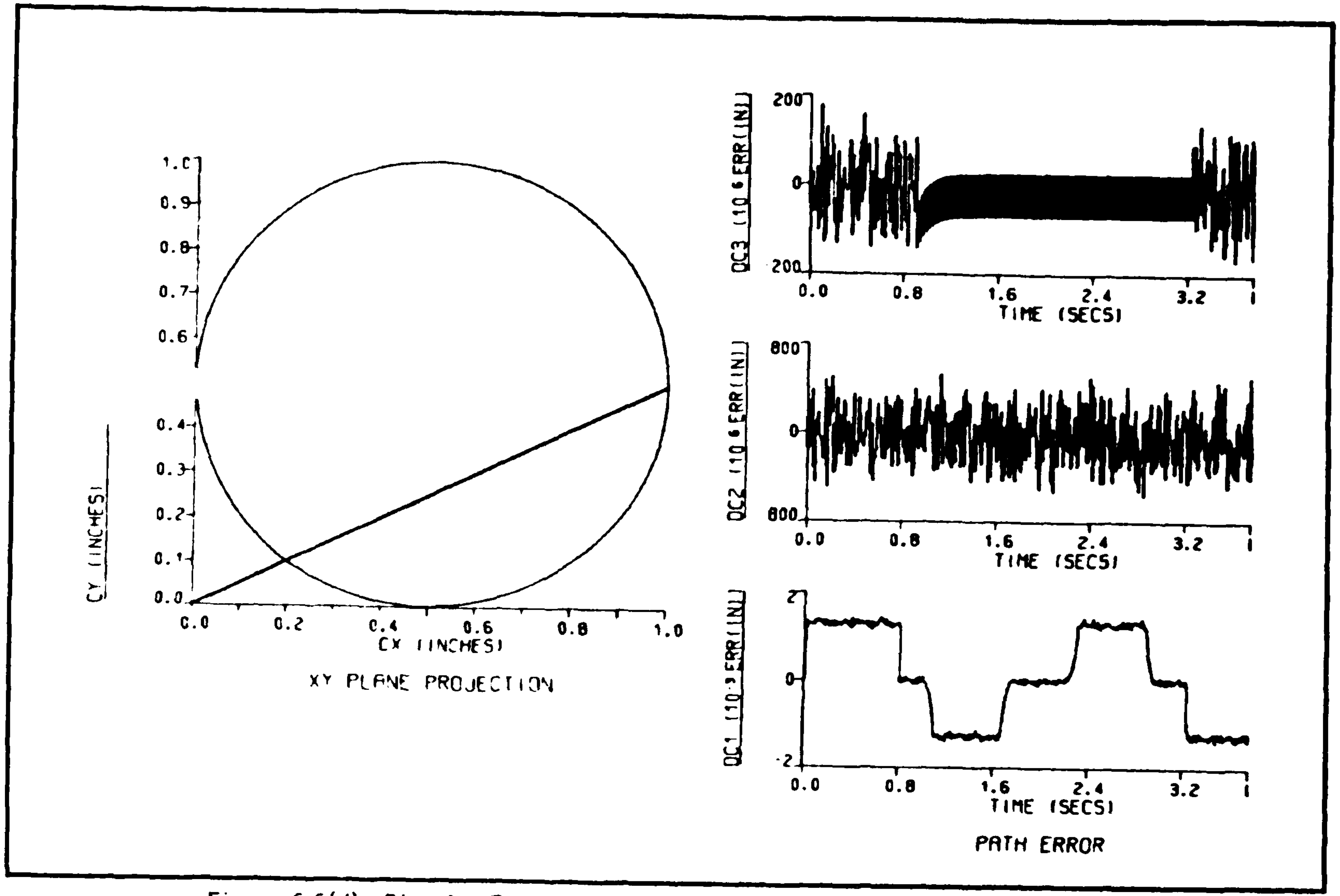


Figure 6.6(d) Circular Path, STEP Excitation, Velocity Method. [T=0.008]. XY Projection and X, Y, and Z Path Error.

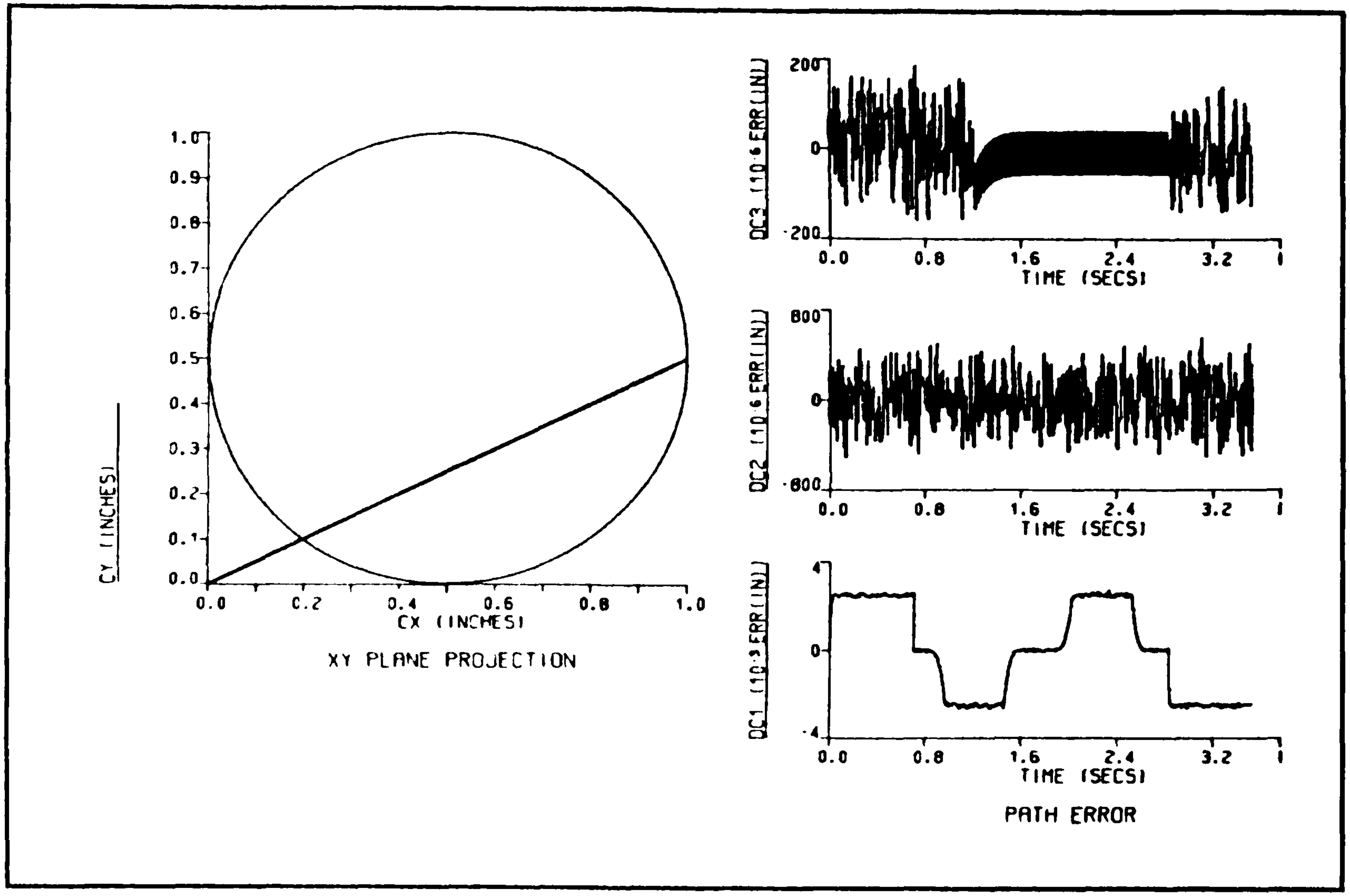


Figure 6.6(e) Circular Path, STEP Excitation, Velocity Method. [T=0.007].
XY Projection and X, Y, and Z Path Error.

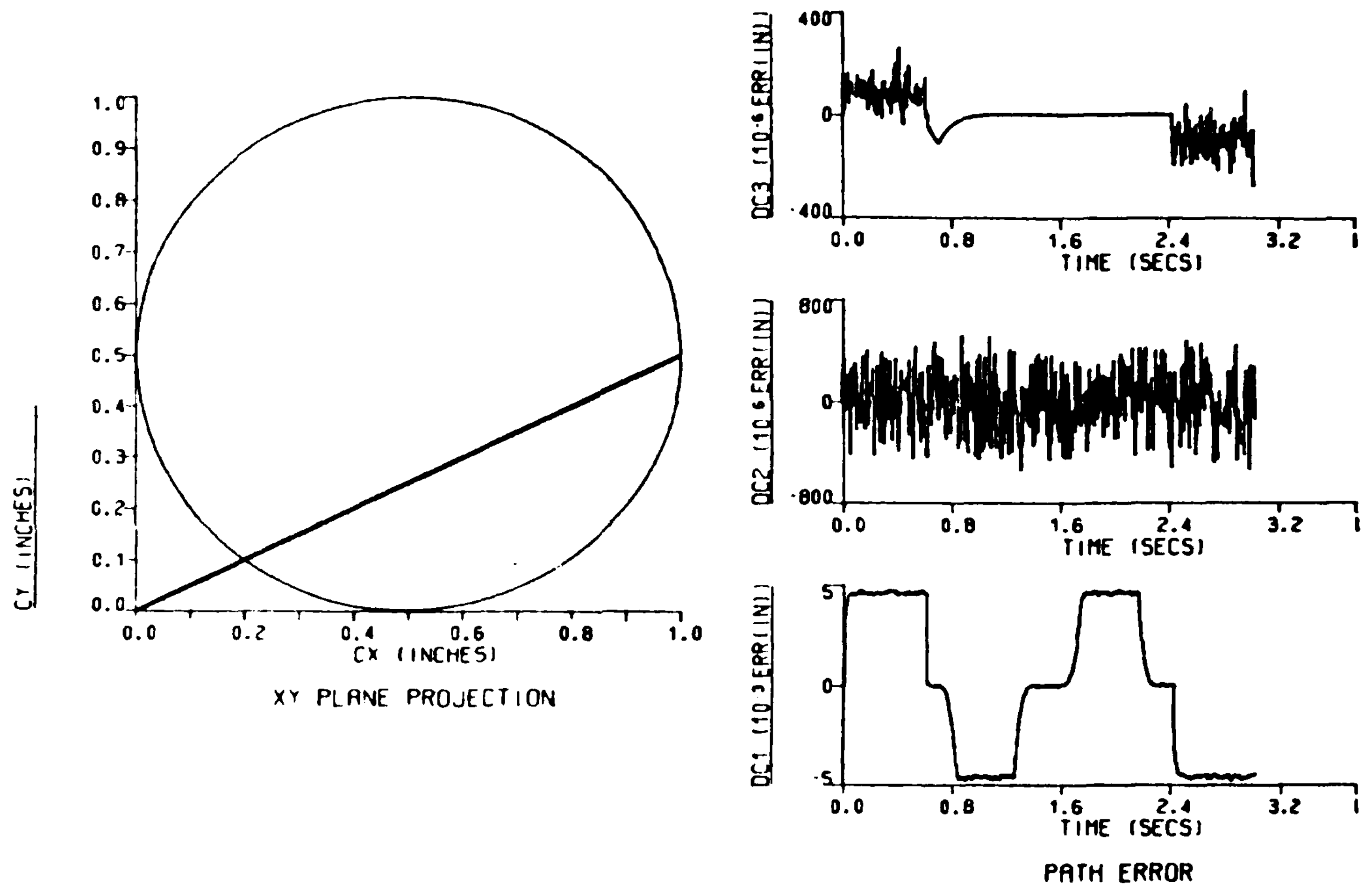


Figure 6.6(f) Circular Path, STEP Excitation, Velocity Method. [T=0.006]. XY Projection and X, Y, and Z Path Error.

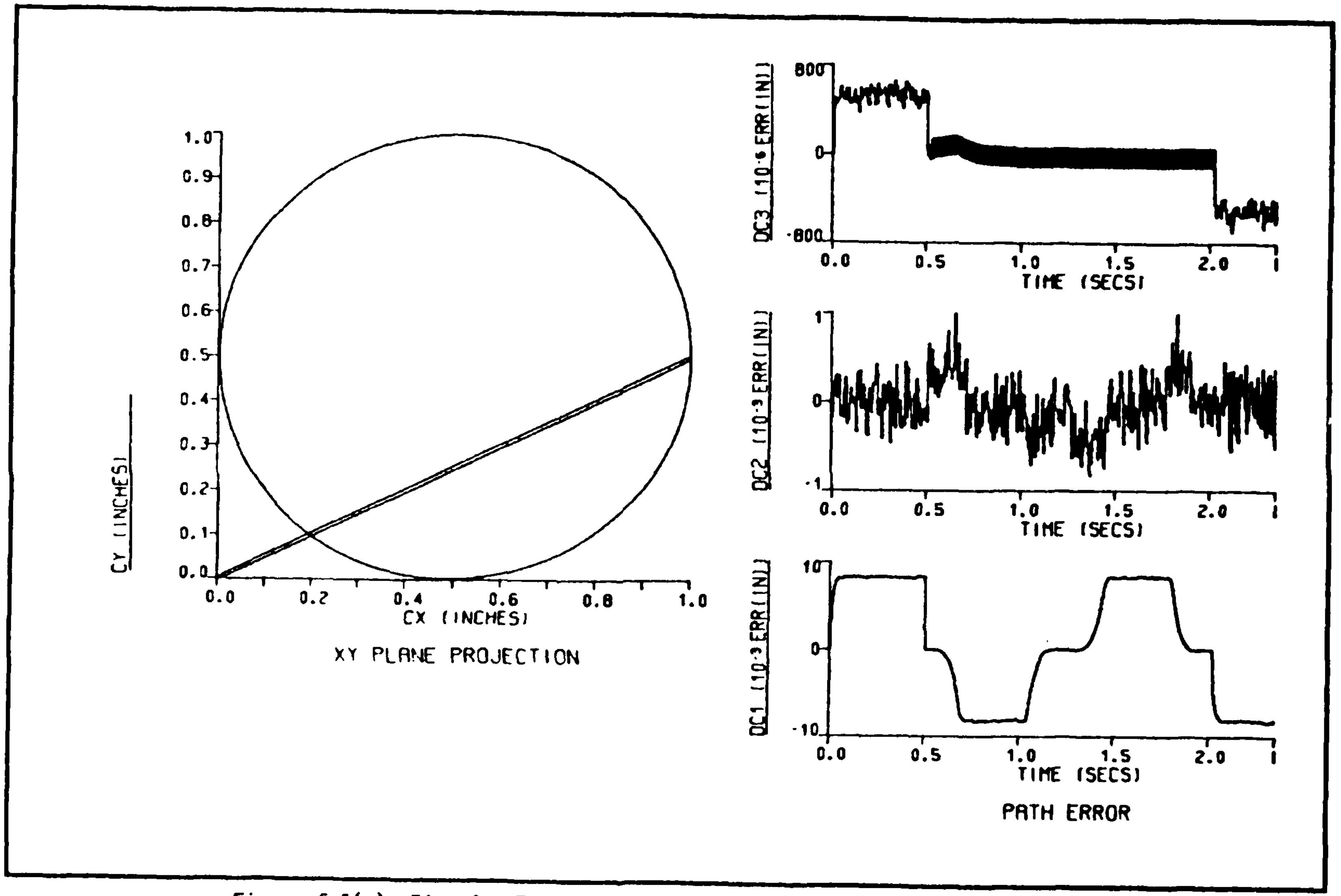


Figure 6.6(g) Circular Path, STEP Excitation, Velocity Method. [T=0.005]. XY Projection and X, Y, and Z Path Error.

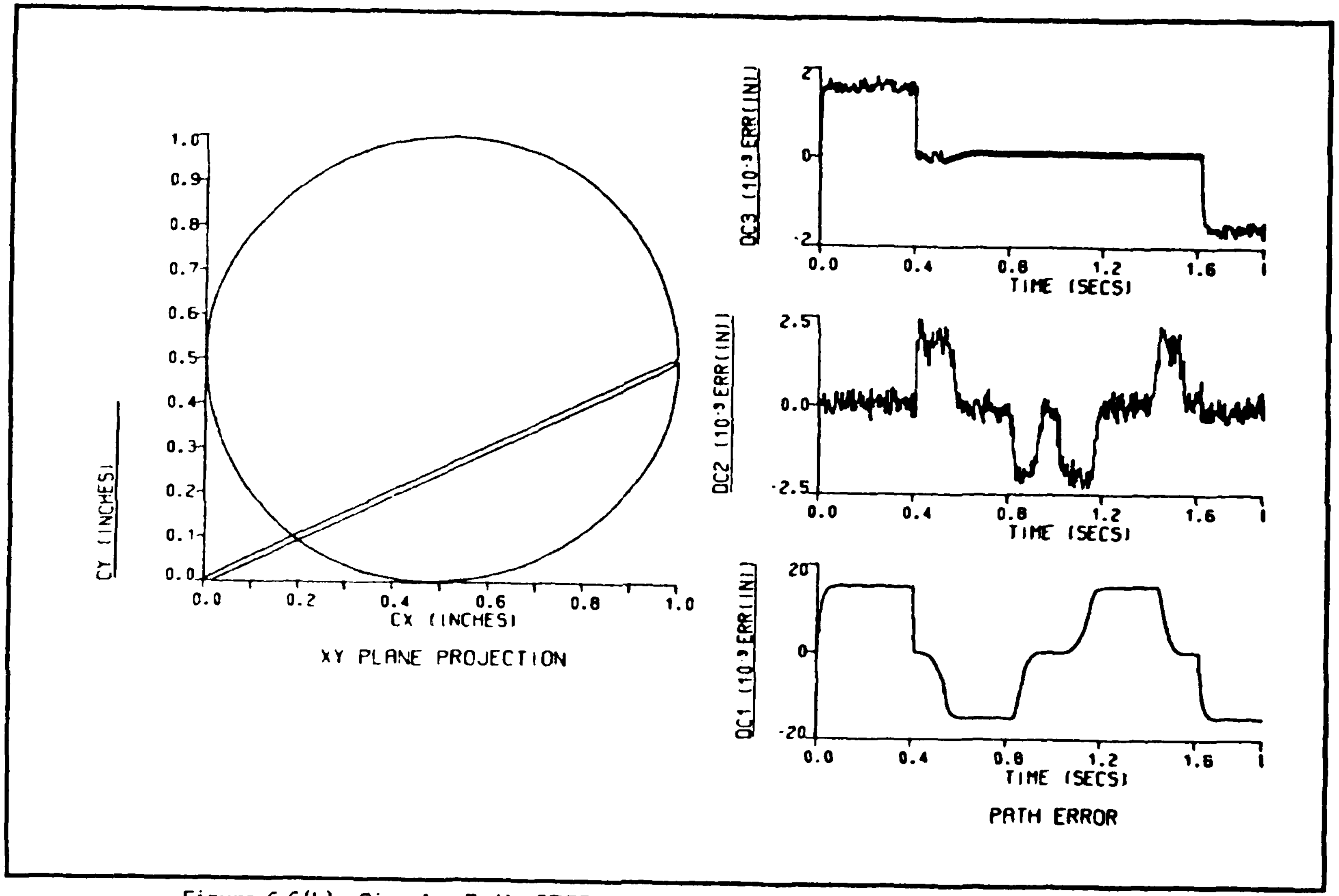


Figure 6.6(h) Circular Path, STEP Excitation, Velocity Method [T=0.004]. XY Projection and X, Y, and Z Path Error.

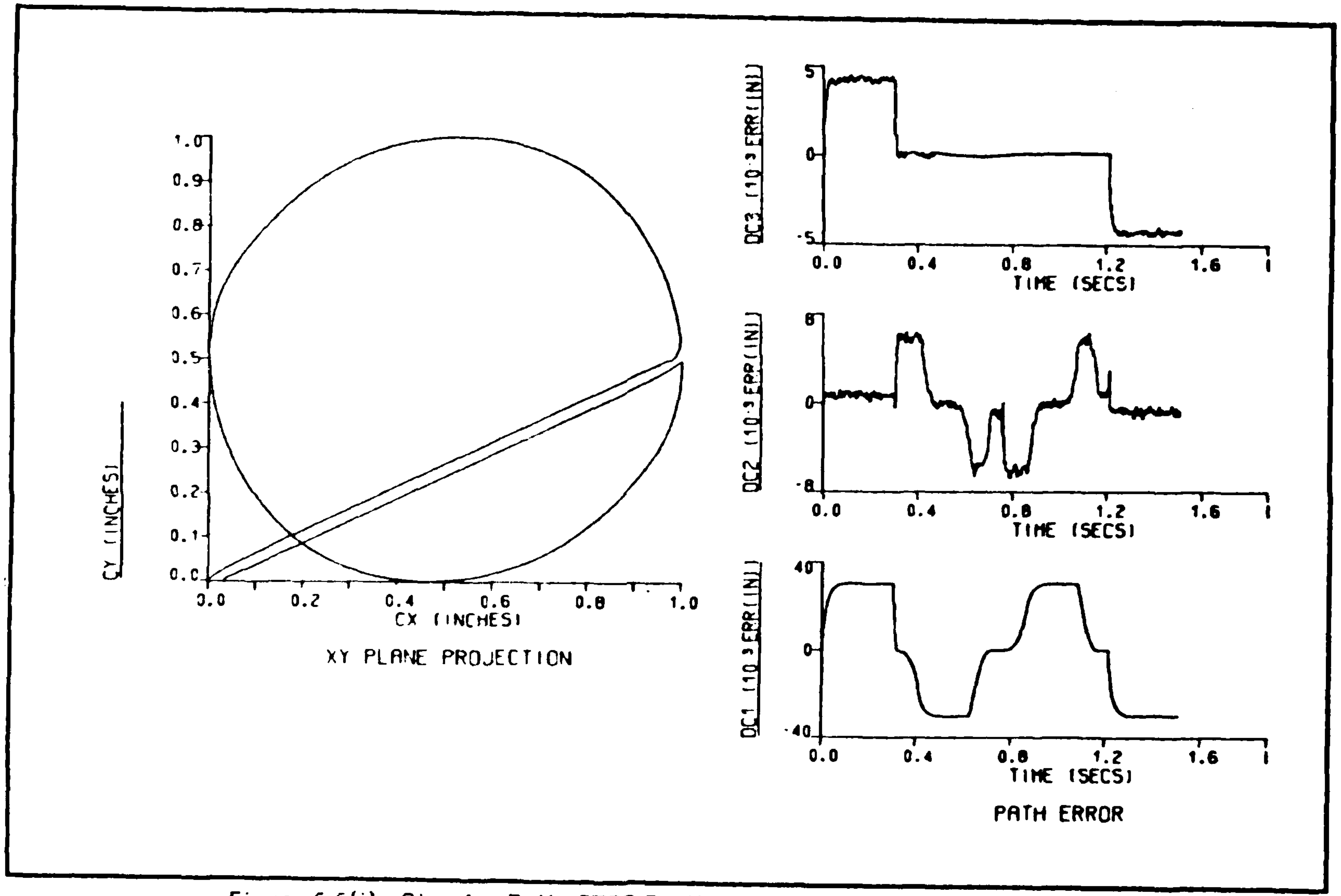


Figure 6.6(i) Circular Path, STEP Excitation, Velocity Method. [T=0.003]. XY Projection and X, Y, and Z Path Error.

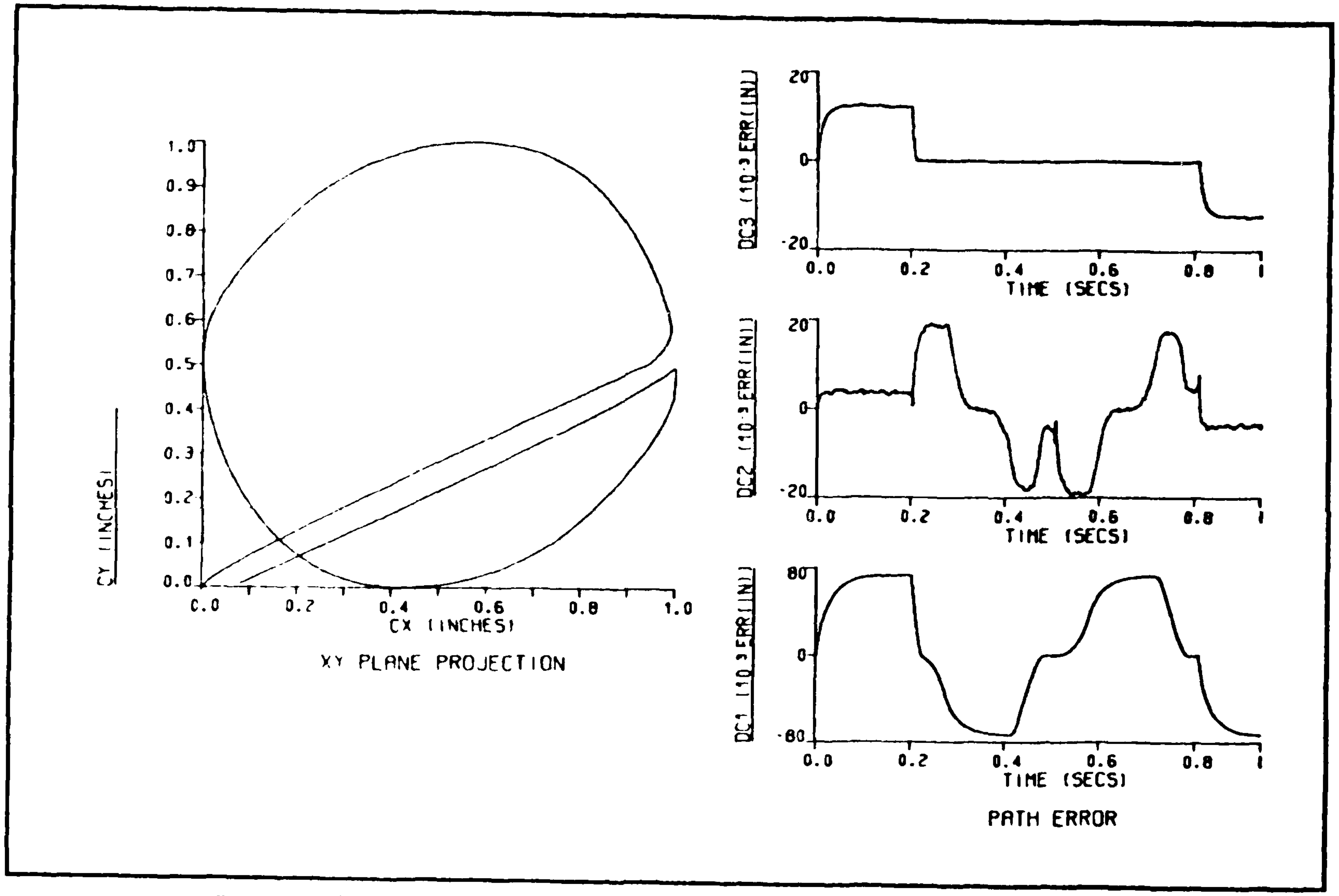
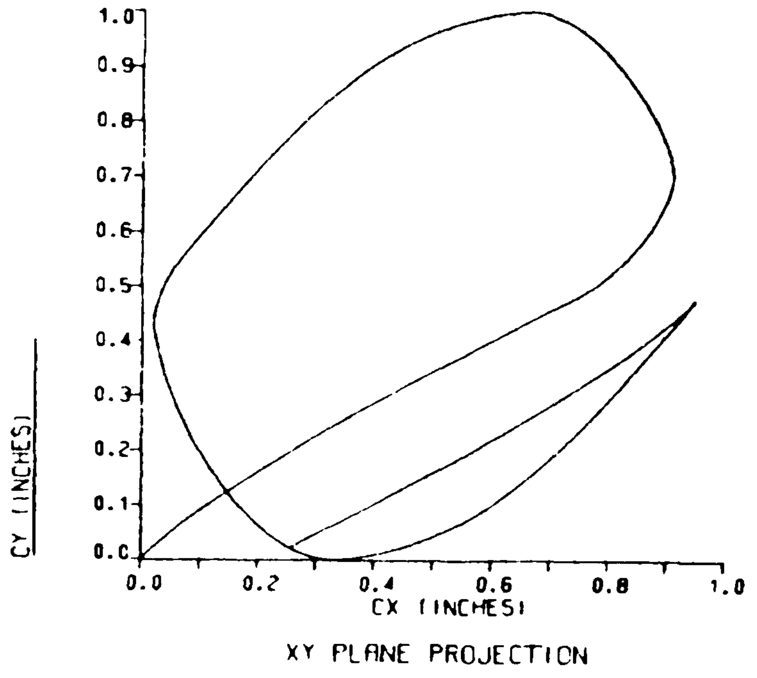


Figure 6 6(j) Circular Path, STEP Excitation, Velocity Method. [T=0.002].
XY Projection and X, Y, and Z Path Error.



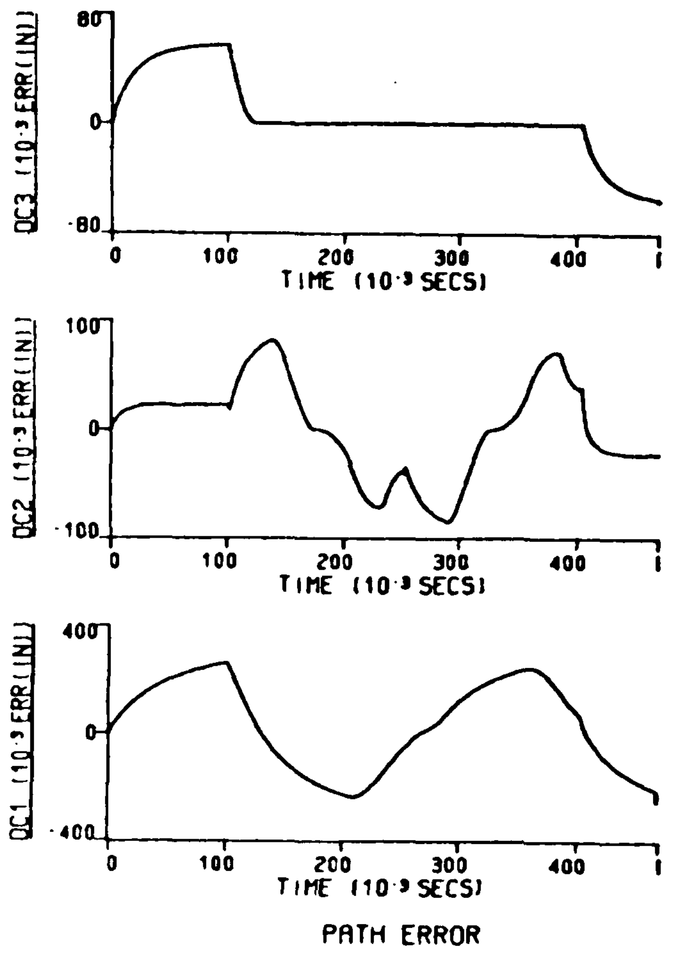


Figure 6.6(k) Circular Path, STEP Excitation, Velocity Method. [T=0.001].
XY Projection and X, Y, and Z Path Error.

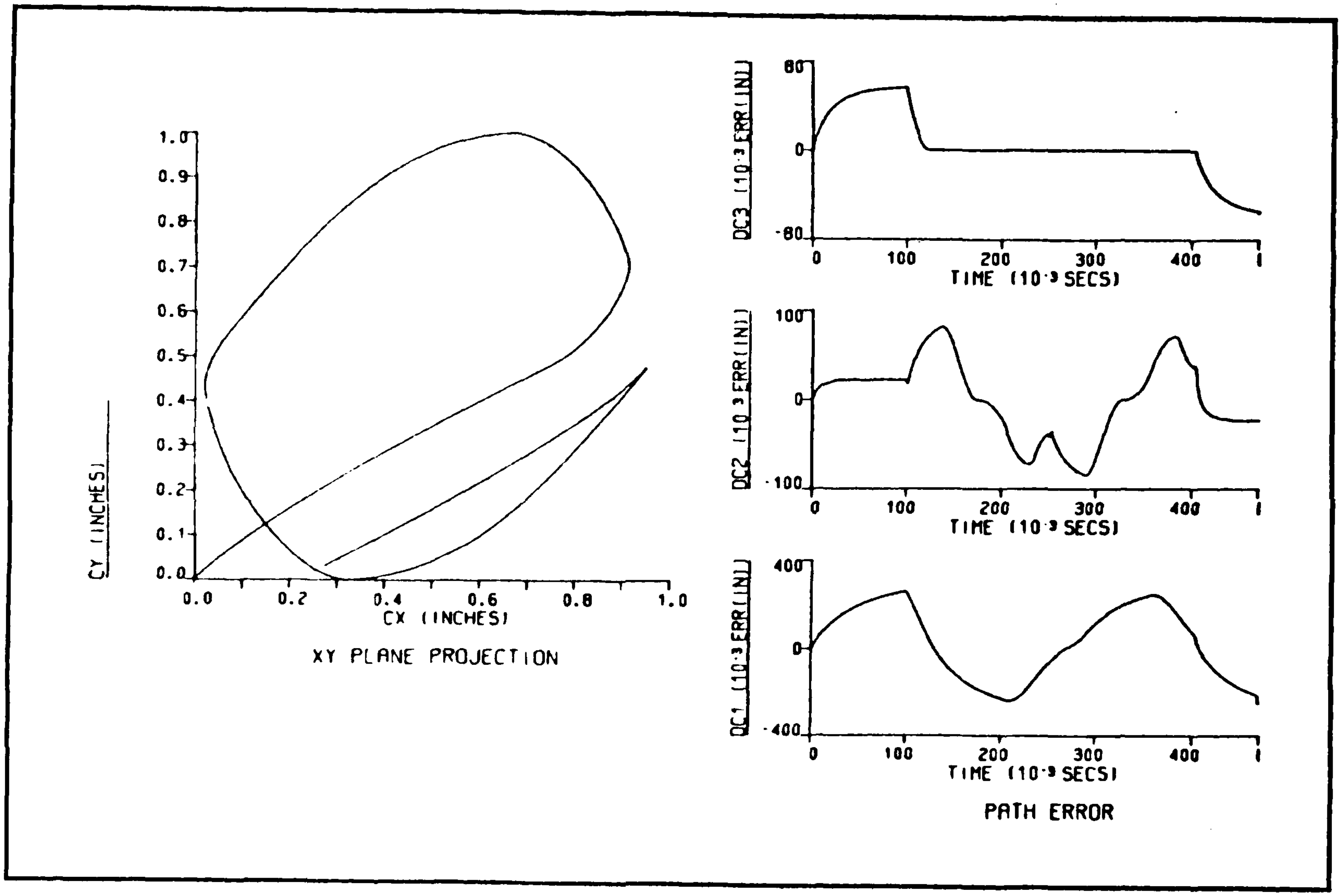


Figure 6.6(l) Circular Path, STEP Excitation, Velocity Method. [T=0.001*].
XY Projection and X, Y, and Z Path Error.

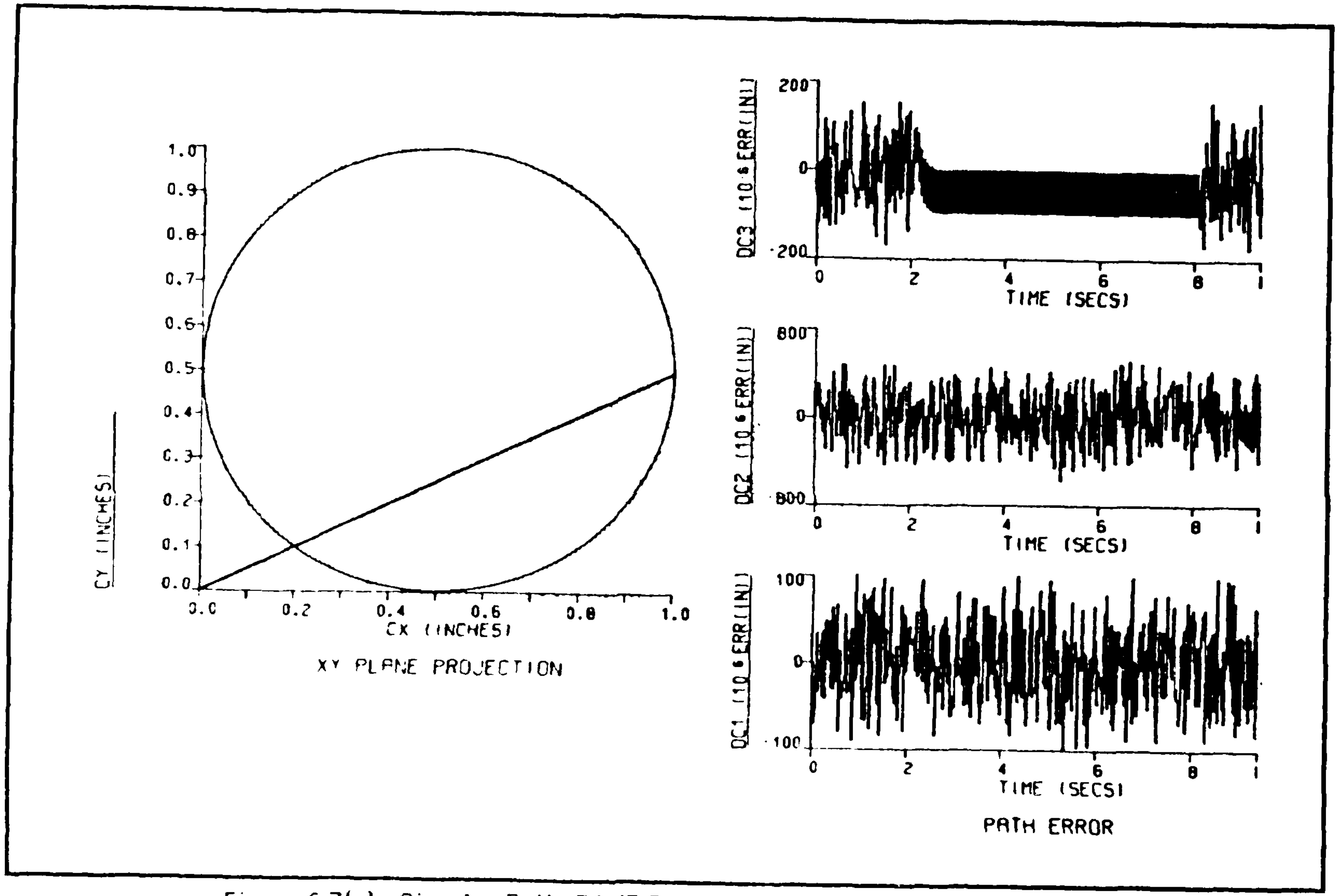


Figure 6.7(a) Circular Path, RAMP Excitation, Velocity Method. [T=0.020]. XY Projection and X, Y, and Z Path Error.

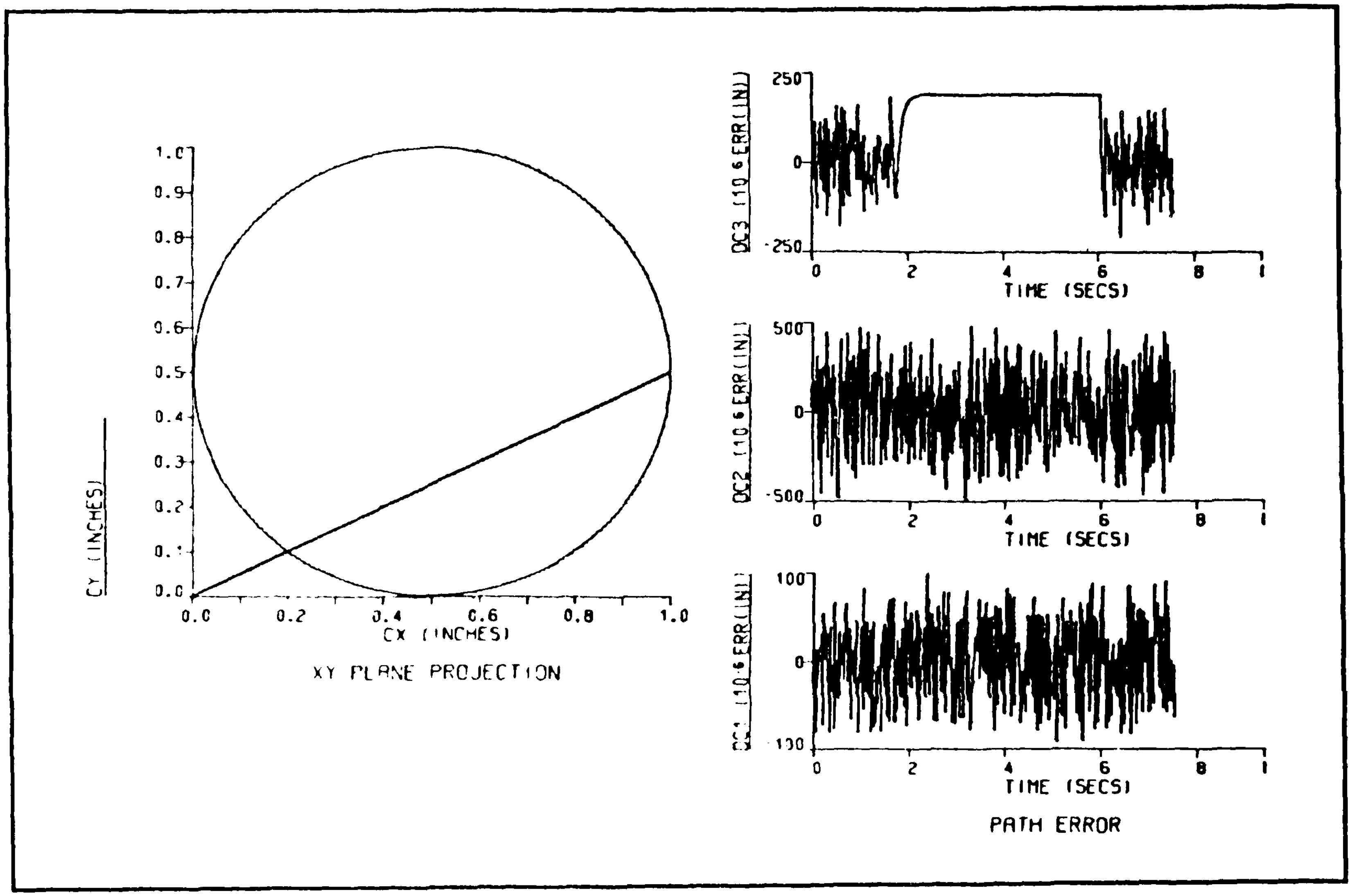


Figure 6.7(b) Circular Path, RAMP Excitation, Velocity Method. [T=0.015]. XY Projection and X, Y, and Z Path Error.

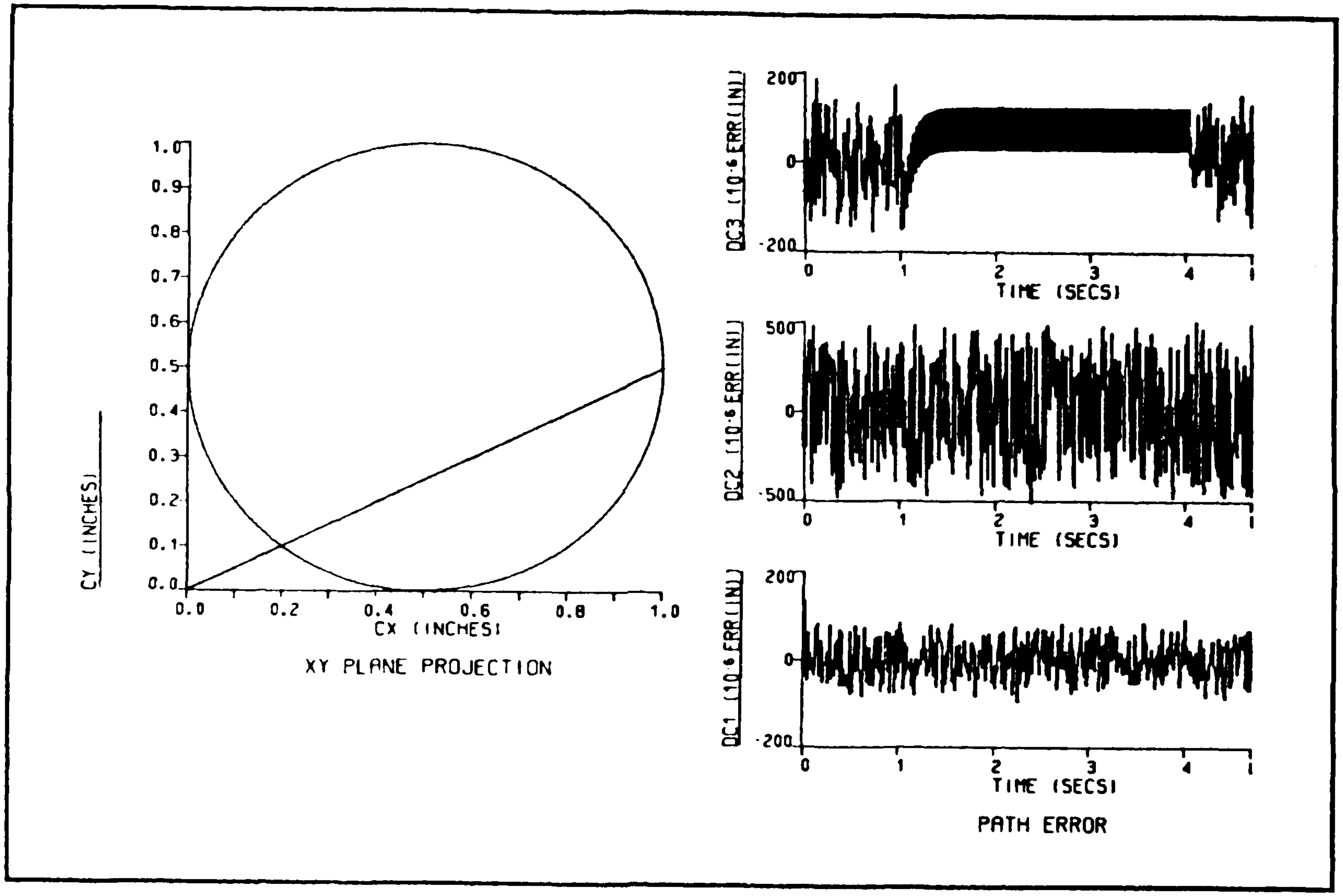


Figure 6.7(c) Circular Path, RAMP Excitation, Velocity Method. [T=0.010].
XY Projection and X, Y, and Z Path Error.

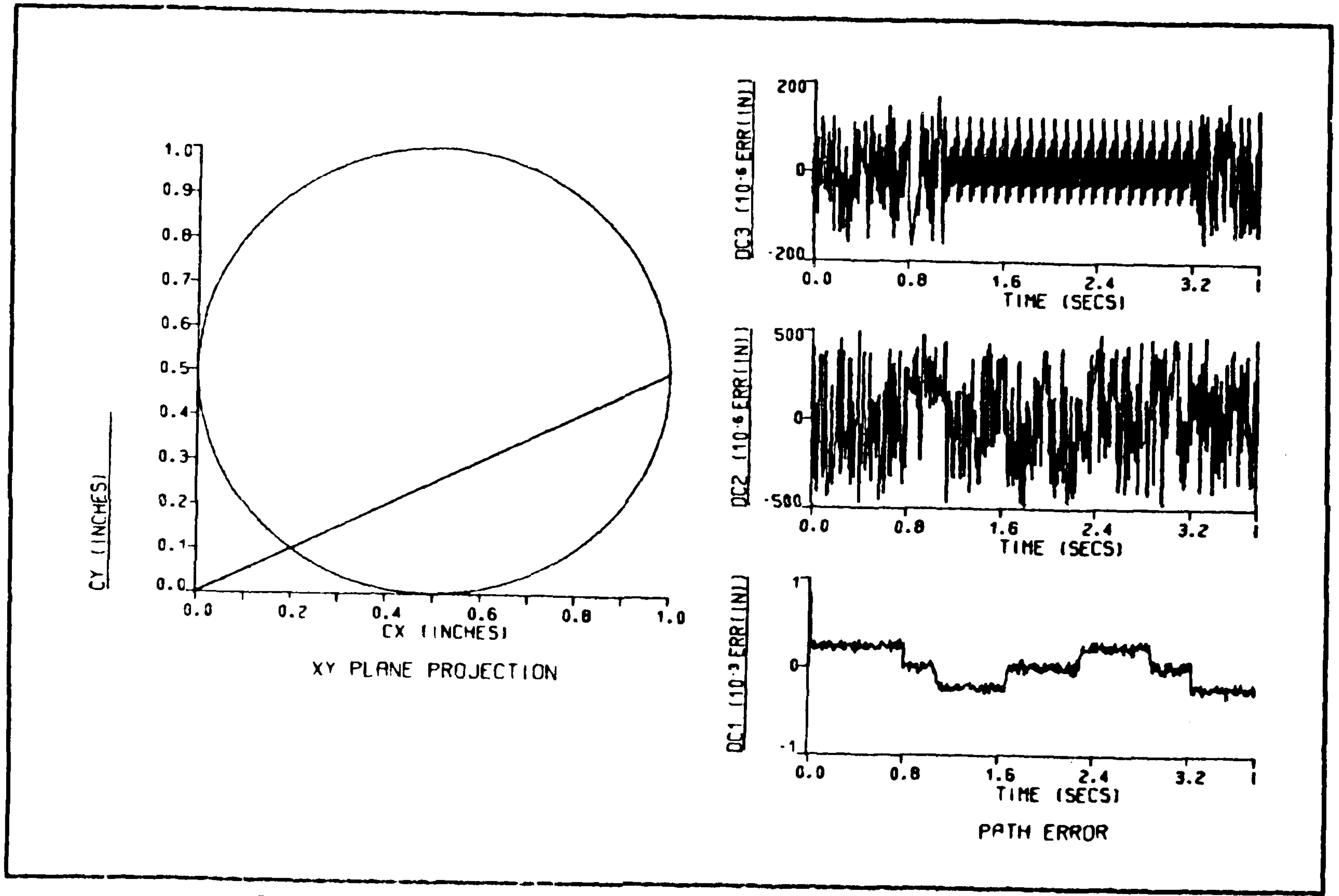


Figure 6.7(d) Circular Path, RAMP Excitation, Velocity Method. [T=0.008]. XY Projection and X, Y, and Z Path Error.

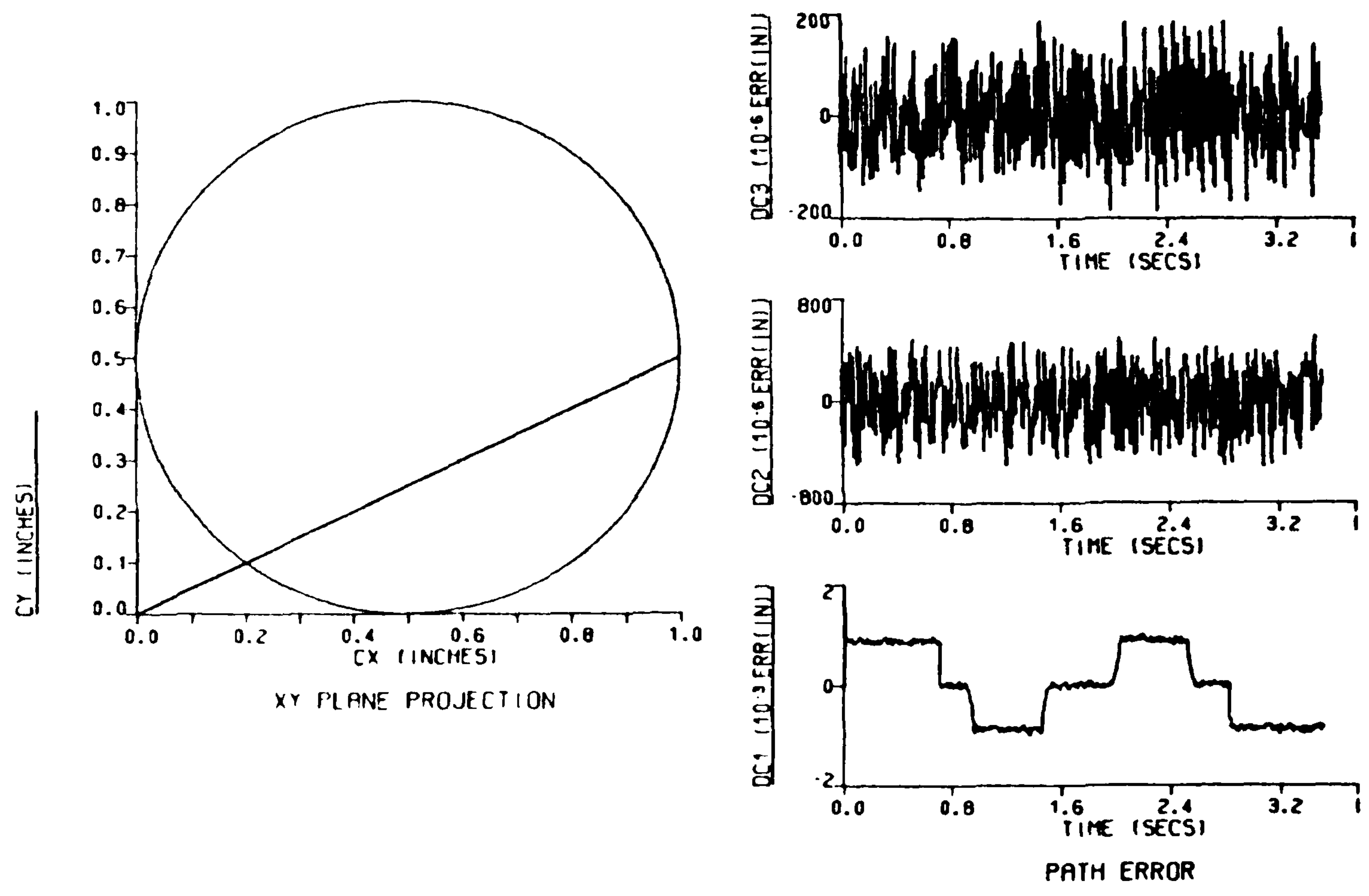


Figure 6.7(e) Circular Path, RAMP Excitation, Velocity Method. [T=0.007].
XY Projection and X, Y, and Z Path Error.

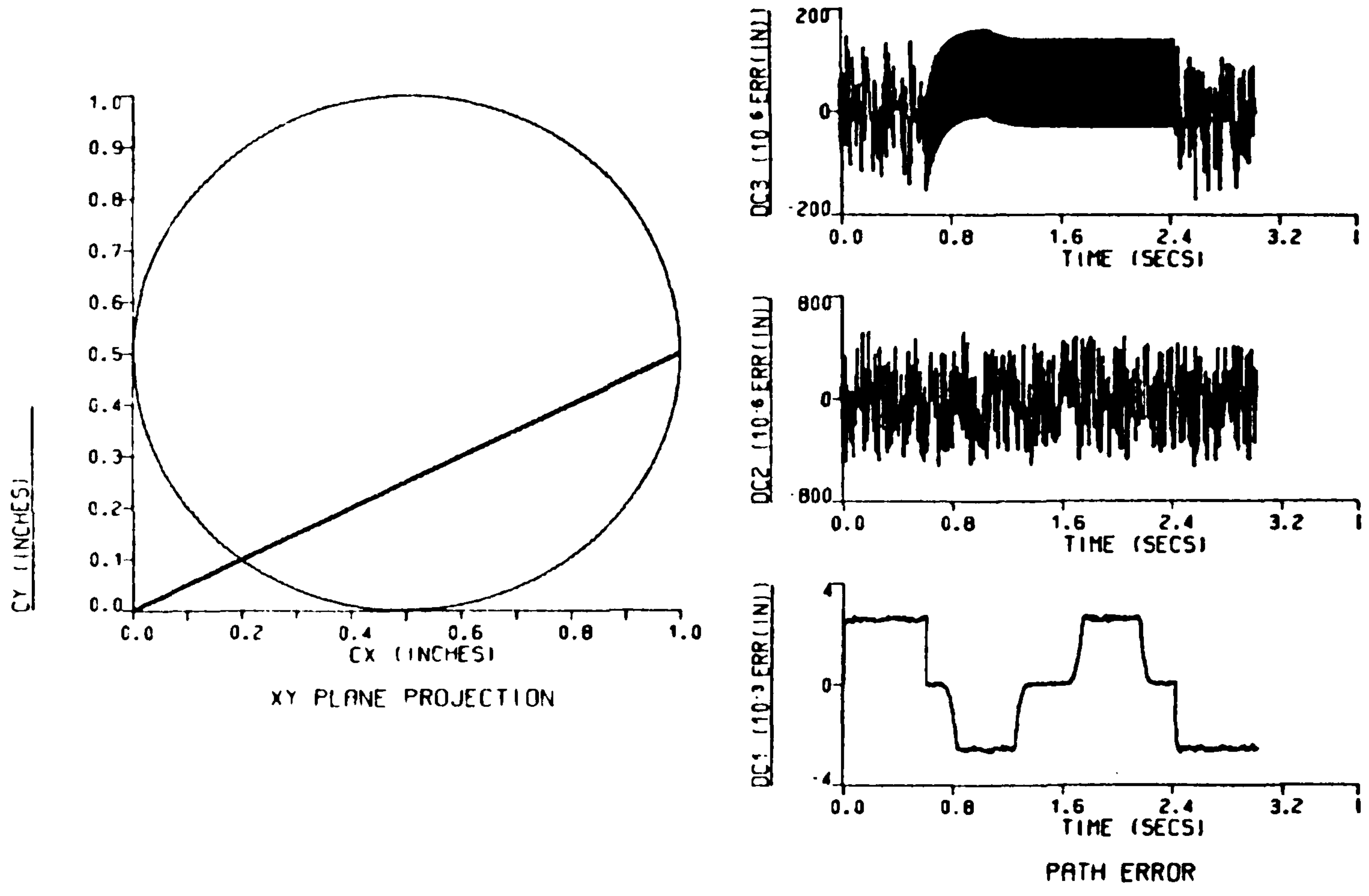


Figure 6.7(f) Circular Path, RAMP Excitation, Velocity Method. [T=0.006]. XY Projection and X, Y, and Z Path Error.

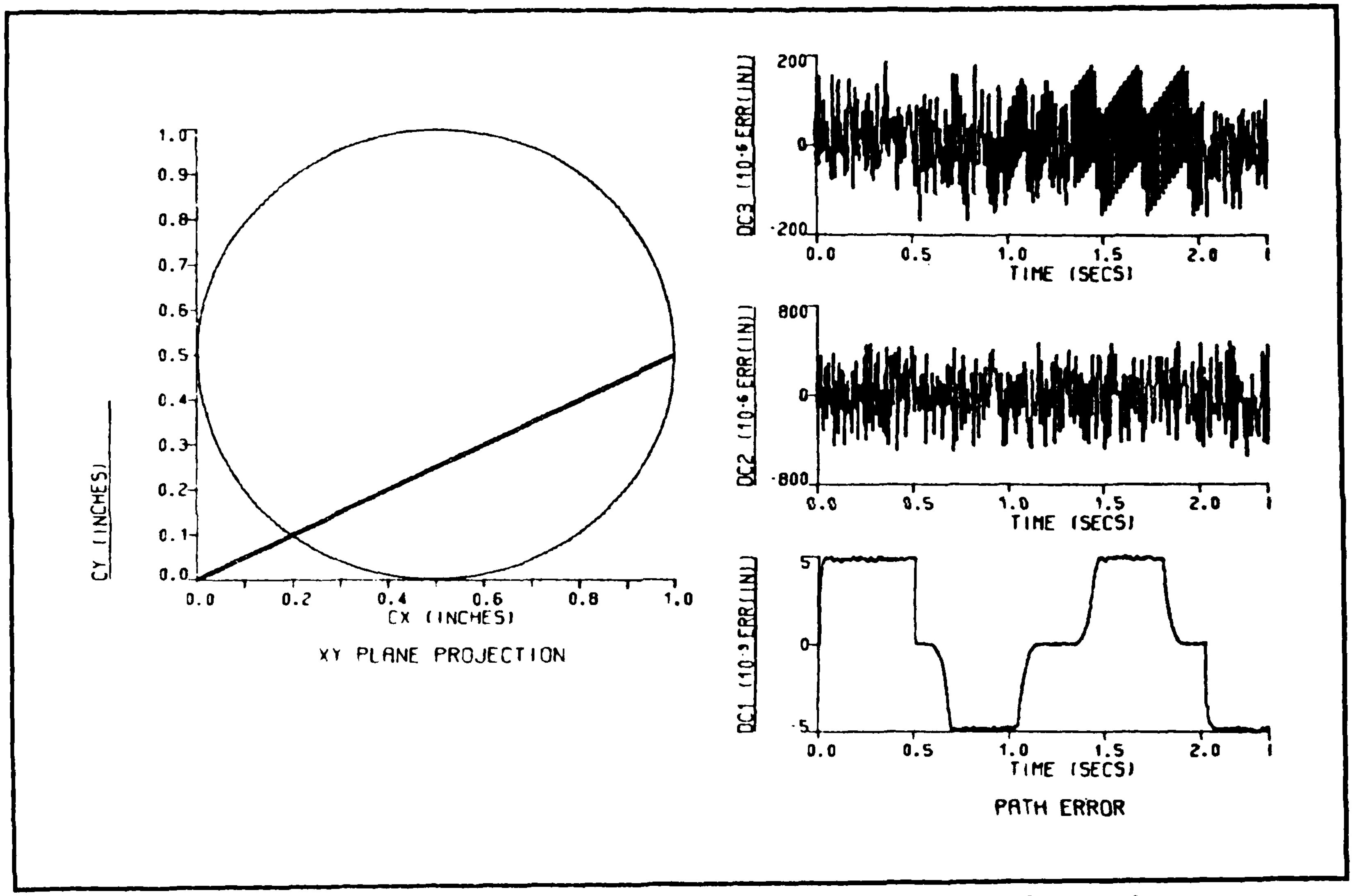


Figure 6.7(g) Circular Path, RAMP Excitation, Velocity Method. [T=0.005].
XY Projection and X, Y, and Z Path Error.

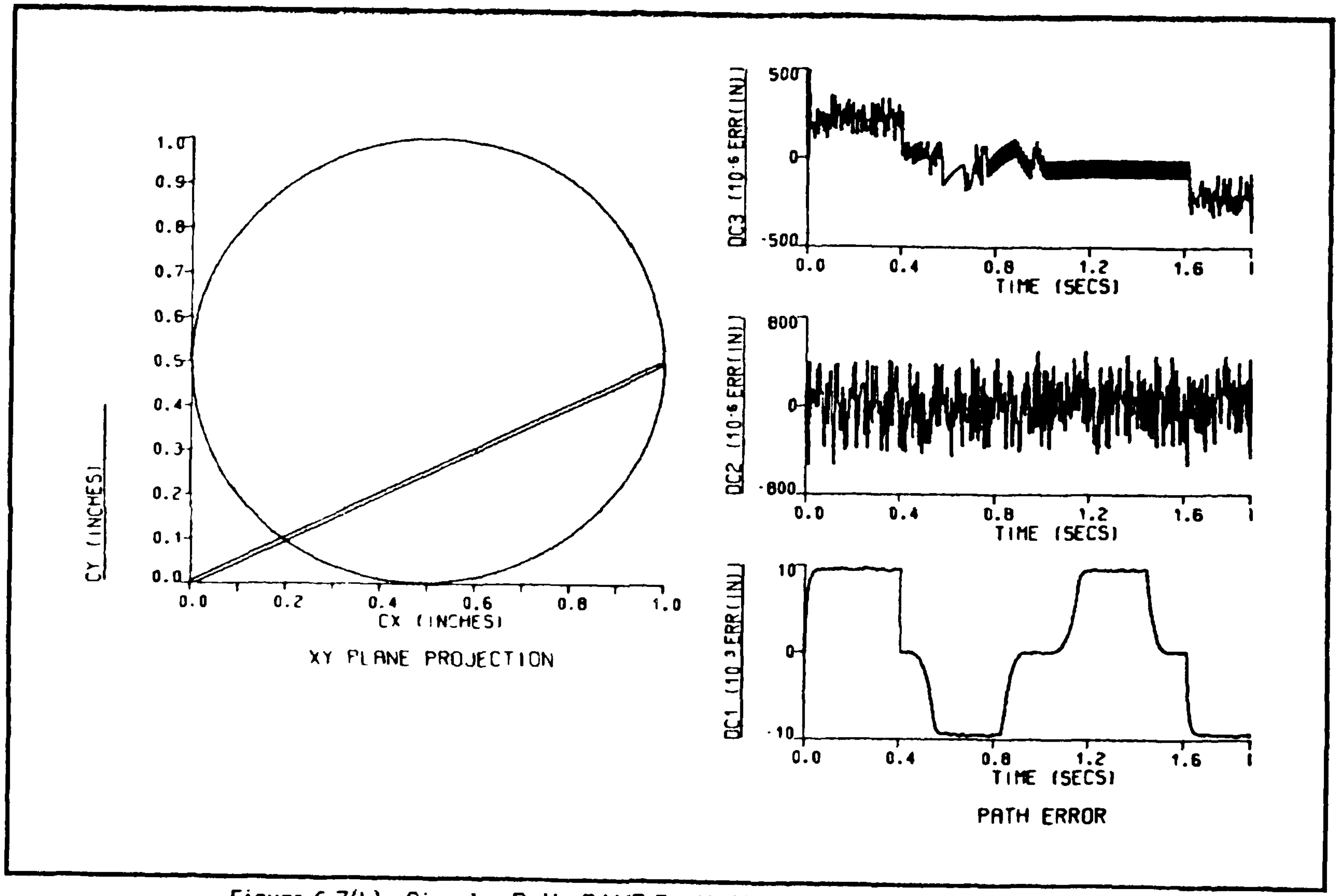


Figure 6 7(h) Circular Path, RAMP Excitation, Velocity Method. [T=0.004].
XY Projection and X, Y, and Z Path Error.

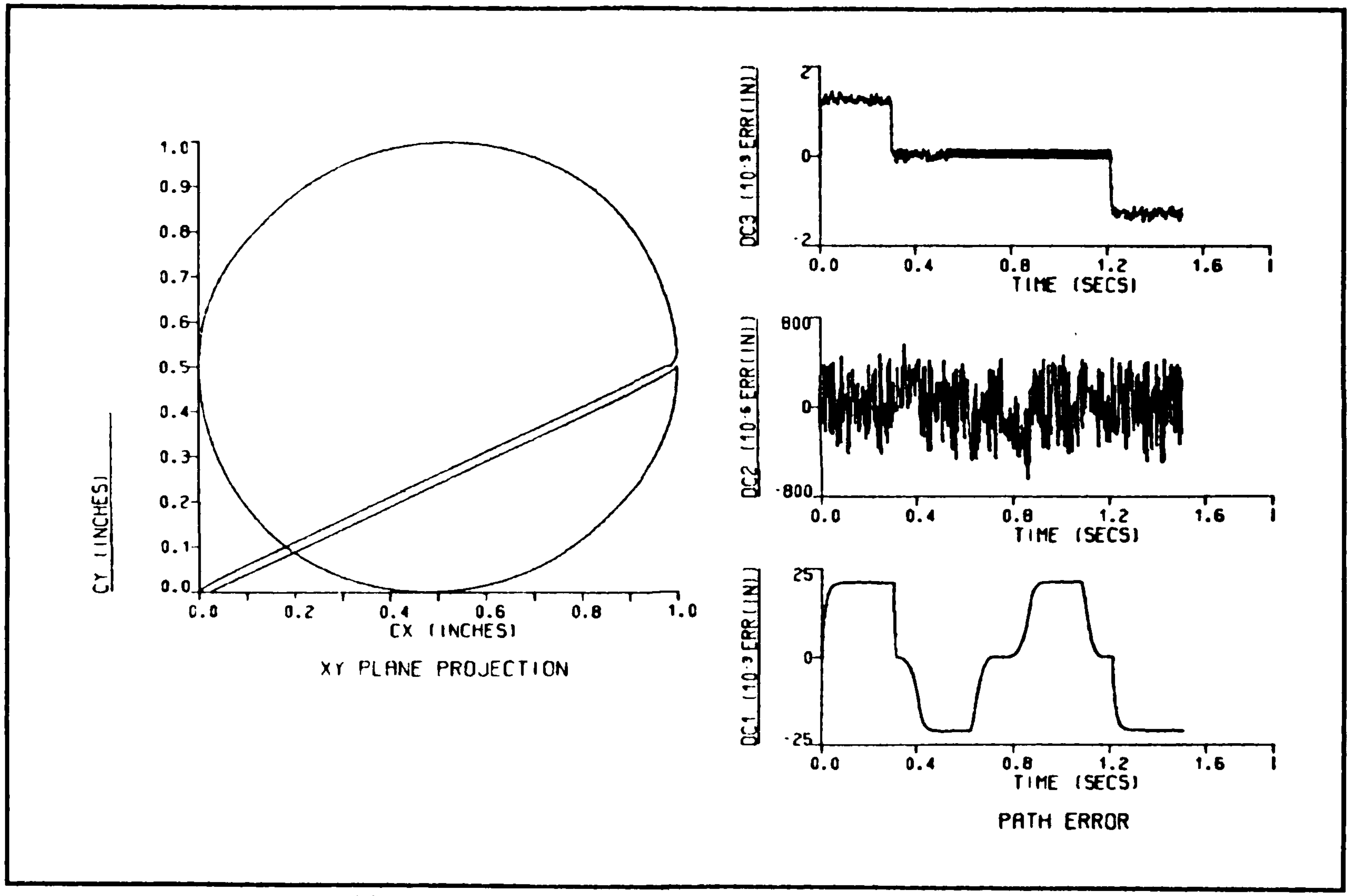


Figure 6.7(i) Circular Path, RAMP Excitation, Velocity Method. [T=0.003]. XY Projection and X, Y, and Z Path Error.

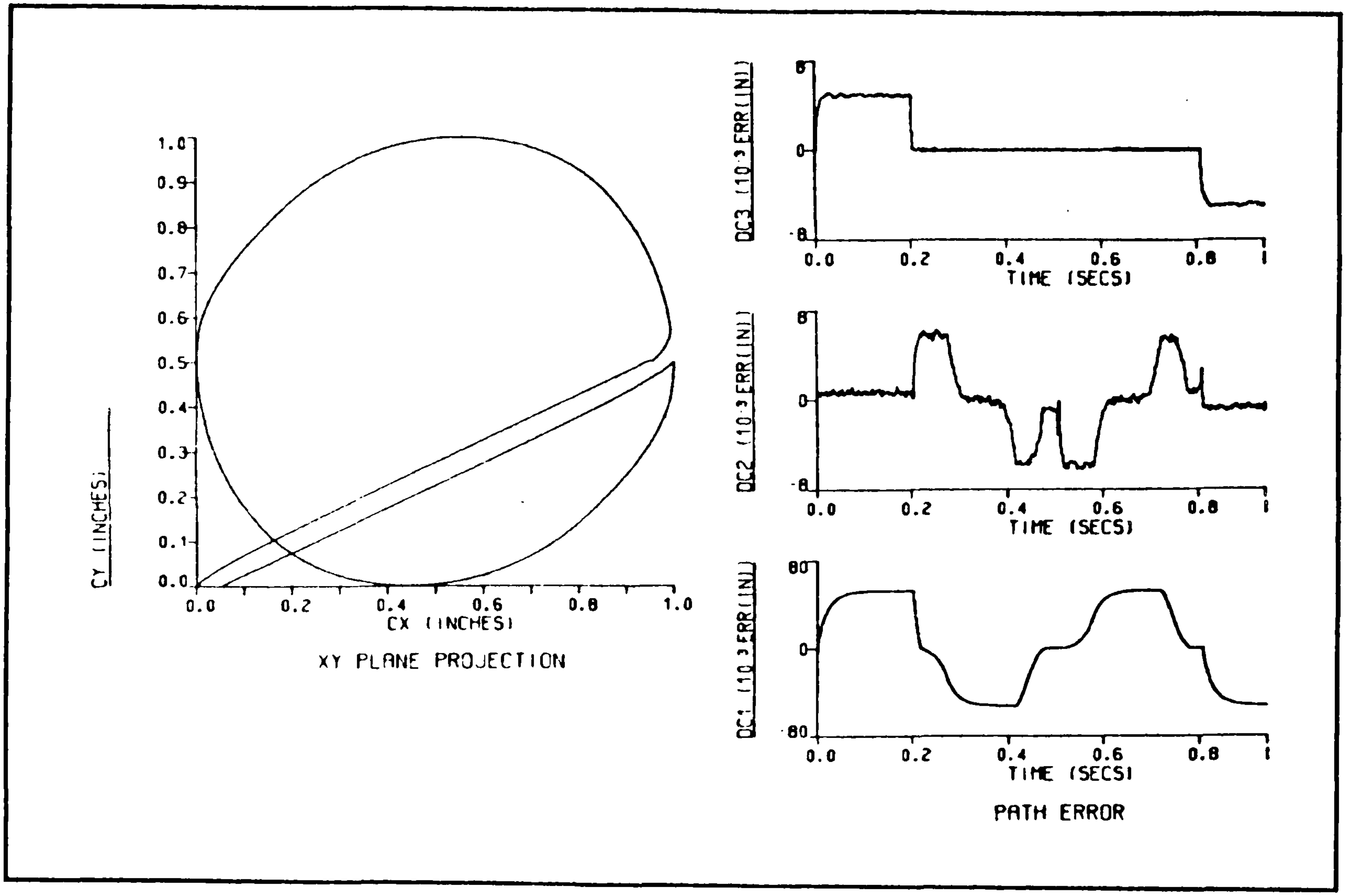


Figure 6.7(j) Circular Path, RAMP Excitation, Velocity Method. [T=0.002]. XY Projection and X, Y, and Z Path Error.

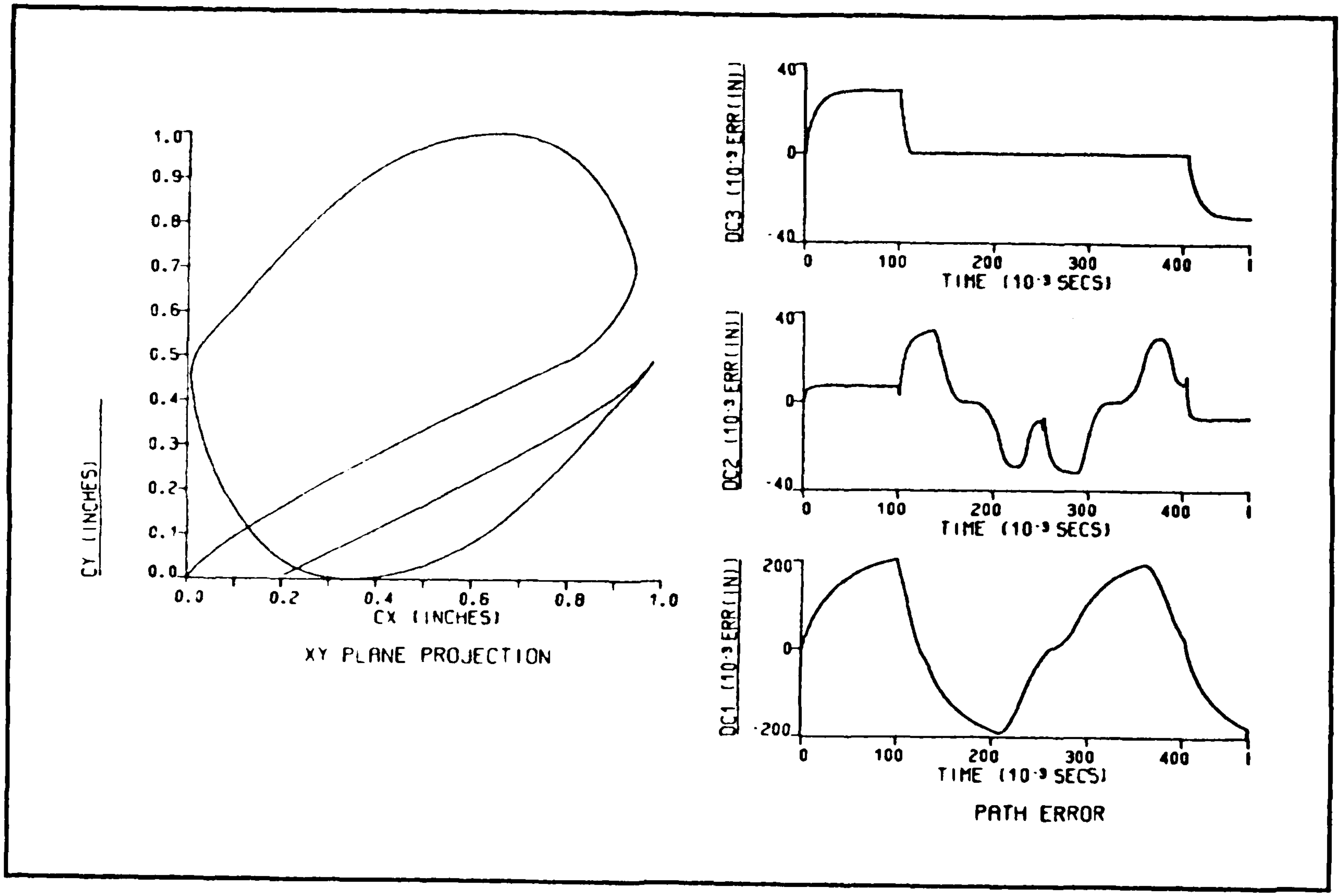
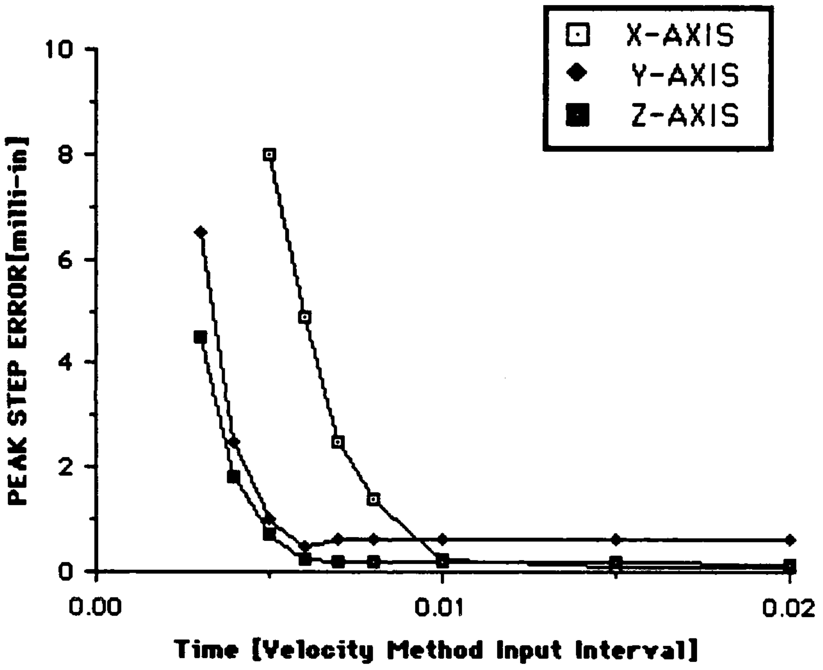
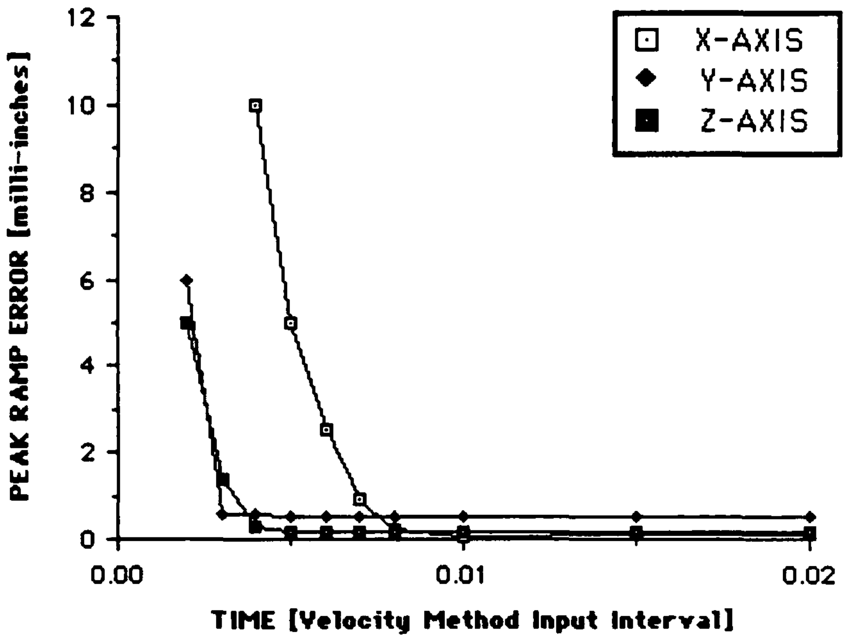


Figure 6.7(k) Circular Path, RAMP Excitation, Velocity Method. [T=0.001]. XY Projection and X, Y, and Z Path Error.



NOTE: LINES BETWEEN CONSECUTIVE POINTS INDICATE ESTIMATE OF VALUES AND SHOW CONTINUITY AND DO NOT INDICATE ACUAL VALUES.

Figure 6.8 (a) Velocity Method STEP Excitation Error.



NOTE: LINES BETWEEN CONSECUTIVE POINTS INDICATE ESTIMATE OF VALUES AND SHOW CONTINUITY AND DO NOT INDICATE ACUAL VALUES.

Figure 6.8 (b) Velocity Method RAMP Excitation Error.

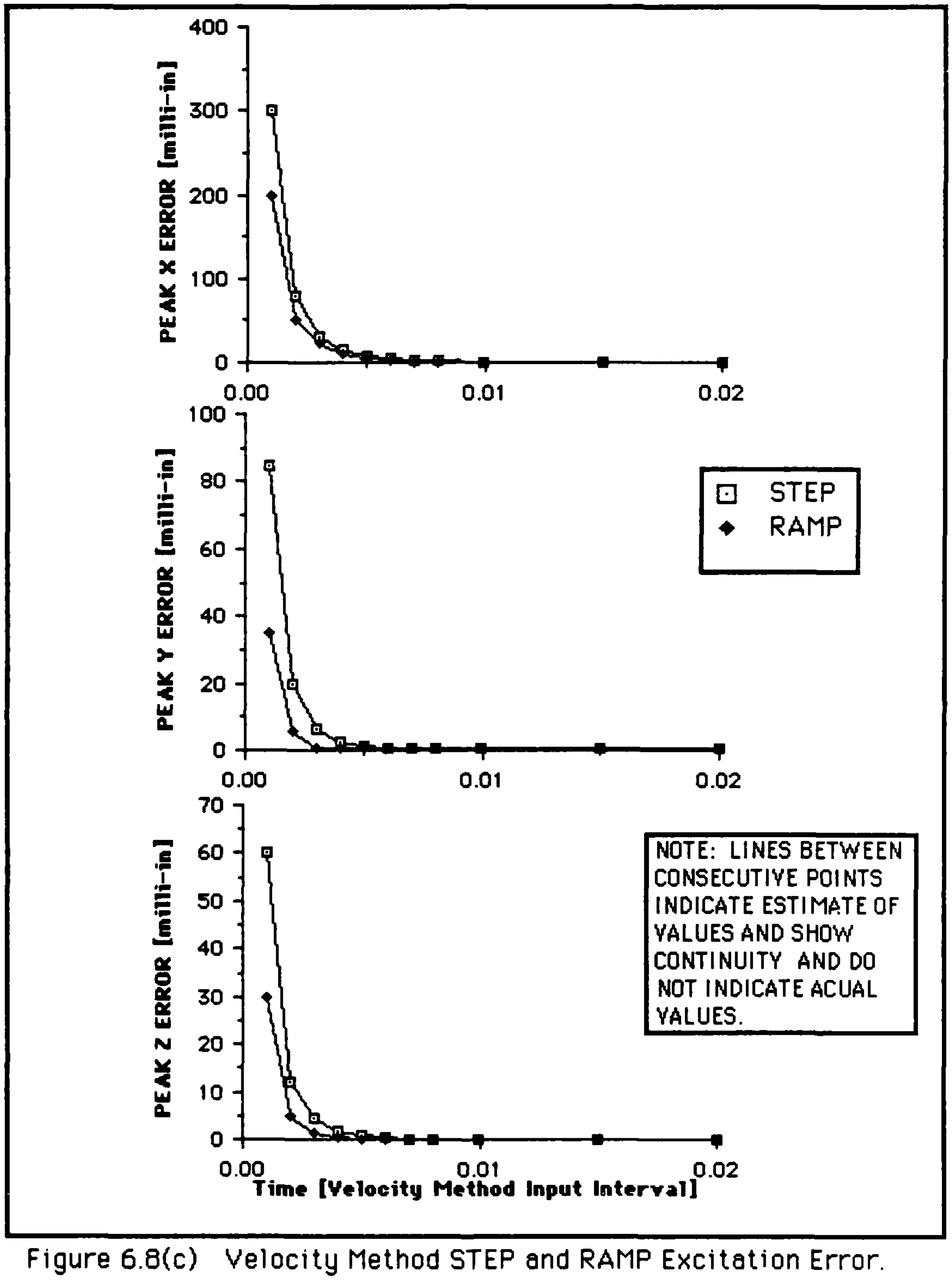


Figure 6.8(c) Velocity Method STEP and RAMP Excitation Error.

Parametric Equations

```
PEQX    =    0.5 * COS ( T ) + 0.5

PEQY    =    0.5 * SIN ( T ) + 0.5

For Helix Path 1

PEQZ    =    T / ( 6 * 3.141529)

For Helix Path 2

PEQZ    =    T / ( 12* 3.141529)

For Both in Equation Bounds Region

        TS      =    0.0

        DELT    =    0.1

    For Helix Path 1

        TF      =    6 * PI

    For Helix Path 2

        TF      =    12*PI
```

Figure 6.9 Parametric Equations and Path For Right Circular Helix's 1 and 2.

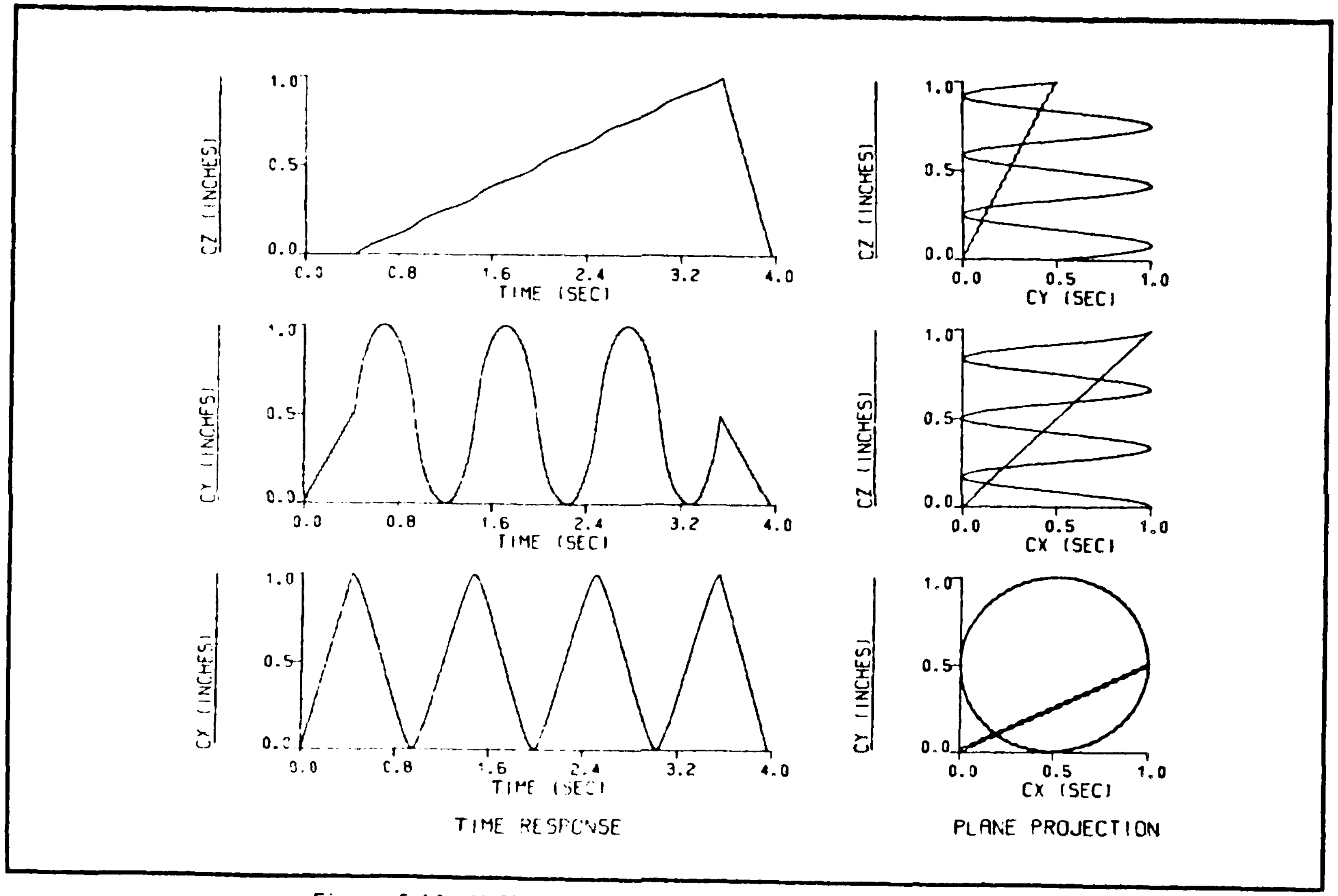


Figure 6.10 Helix Path 1, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response (b) XY, XZ, and YZ Plane Projection.

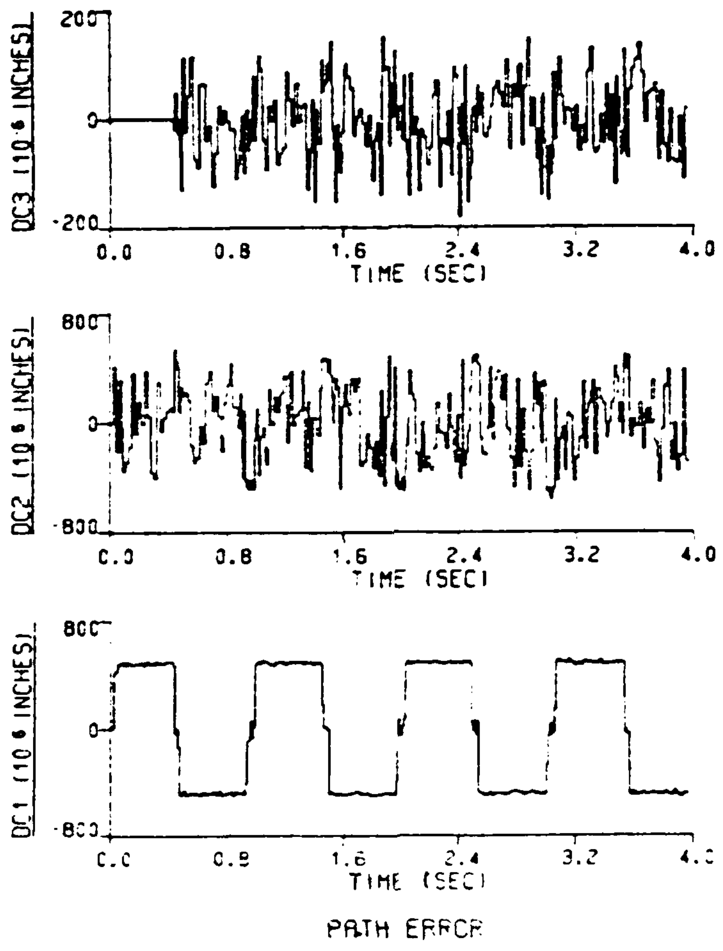


Figure 6.10(c) Helix Path 1, STEP Excitation, Position Method.
X, Y, and Z Path Error.

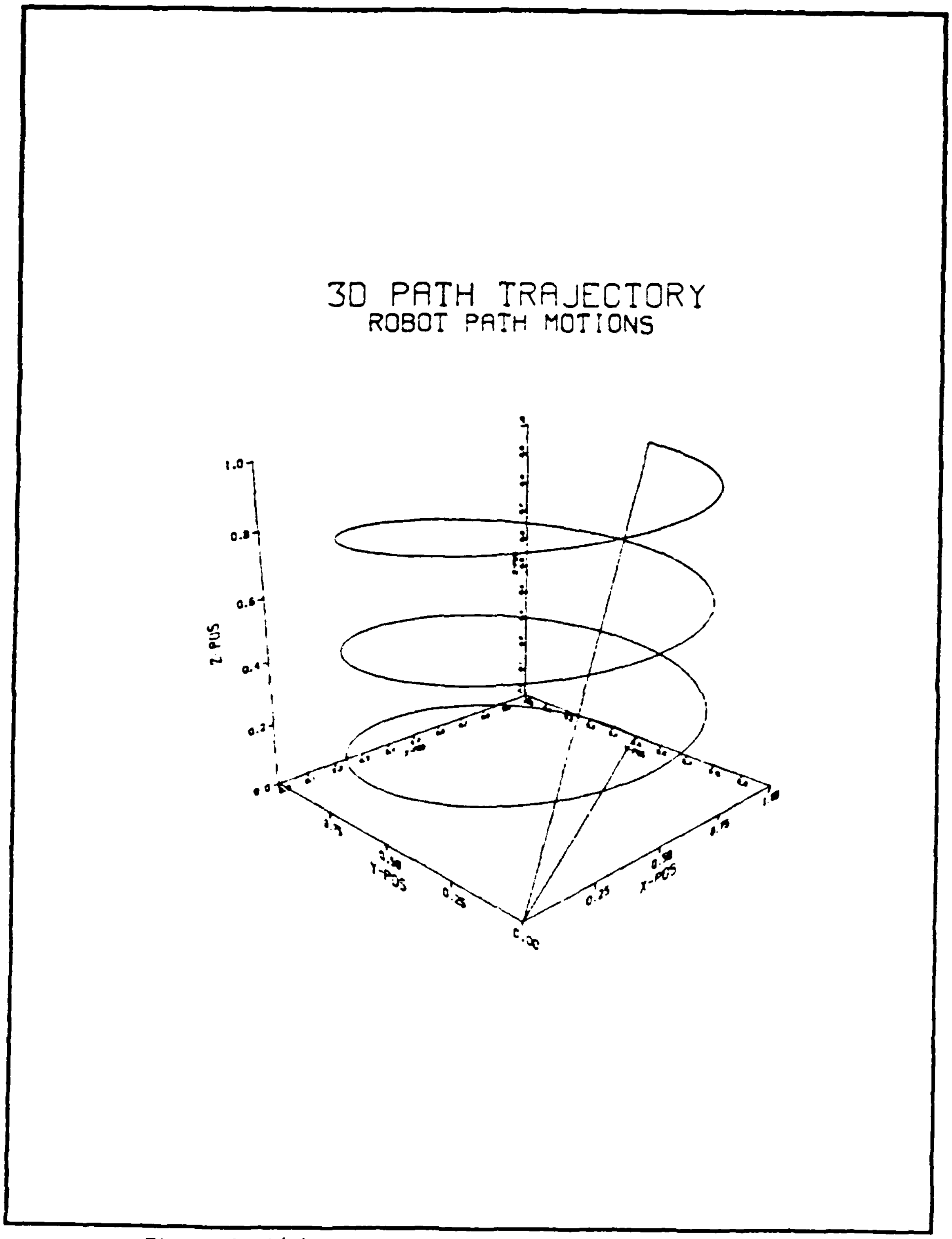


Figure 6.10(d) 3-Dimensional View of Helix Path 1.

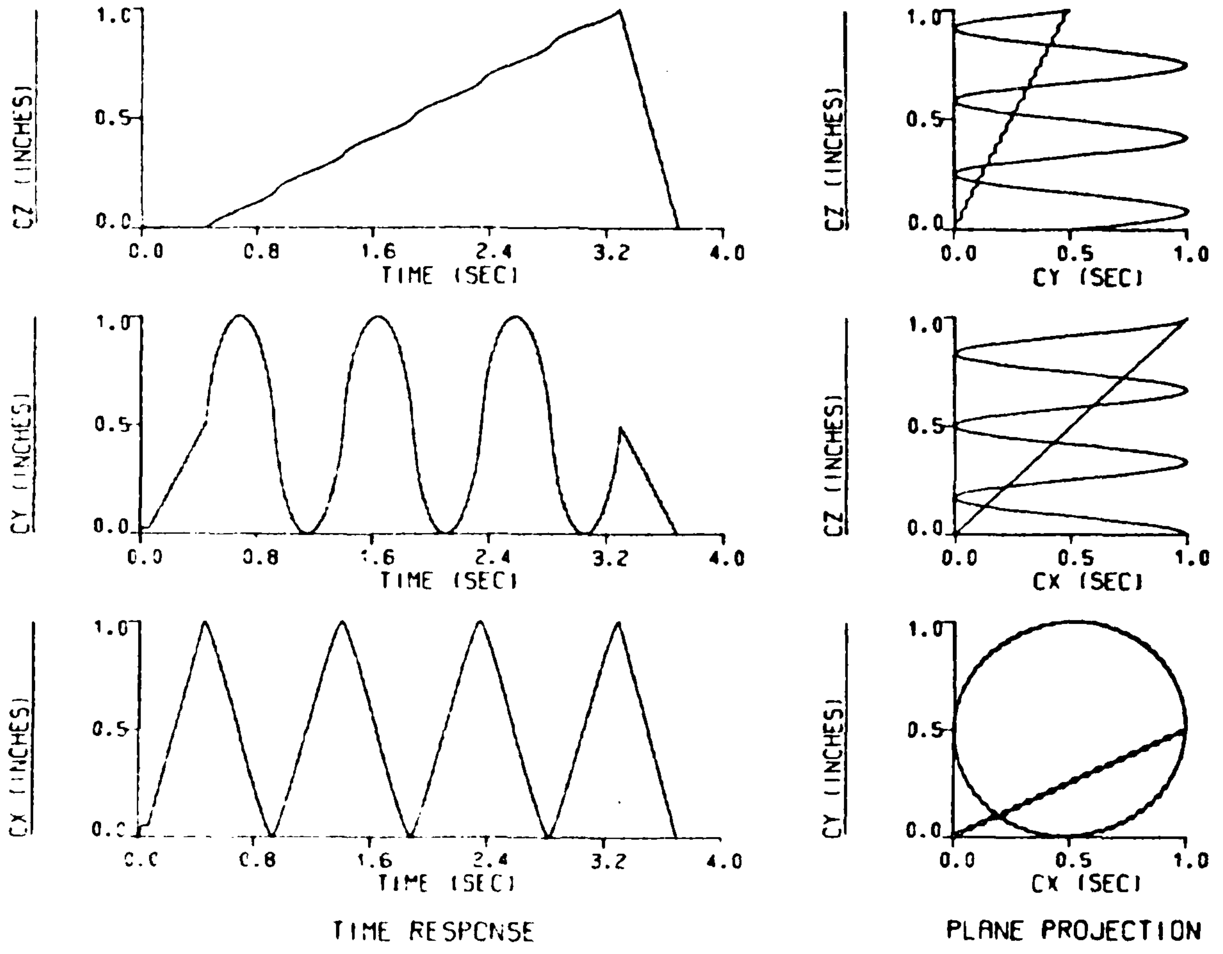


Figure 6.11 Helix Path 1, RAMP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

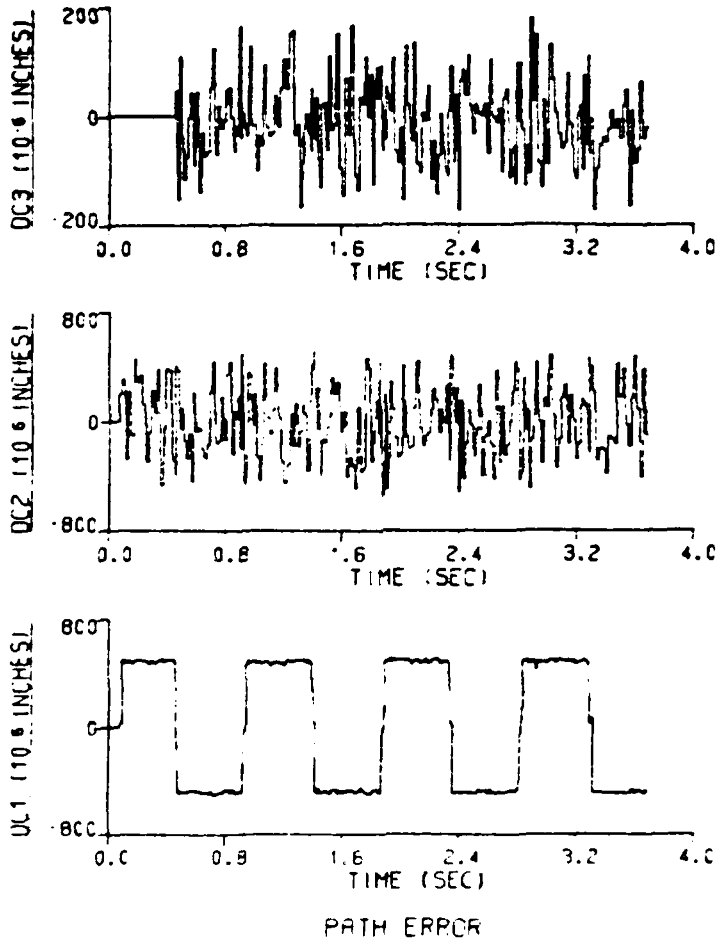


Figure 6.11(c) Helix Path 1, RAMP Excitation, Position Method. X, Y, and Z Path Error.

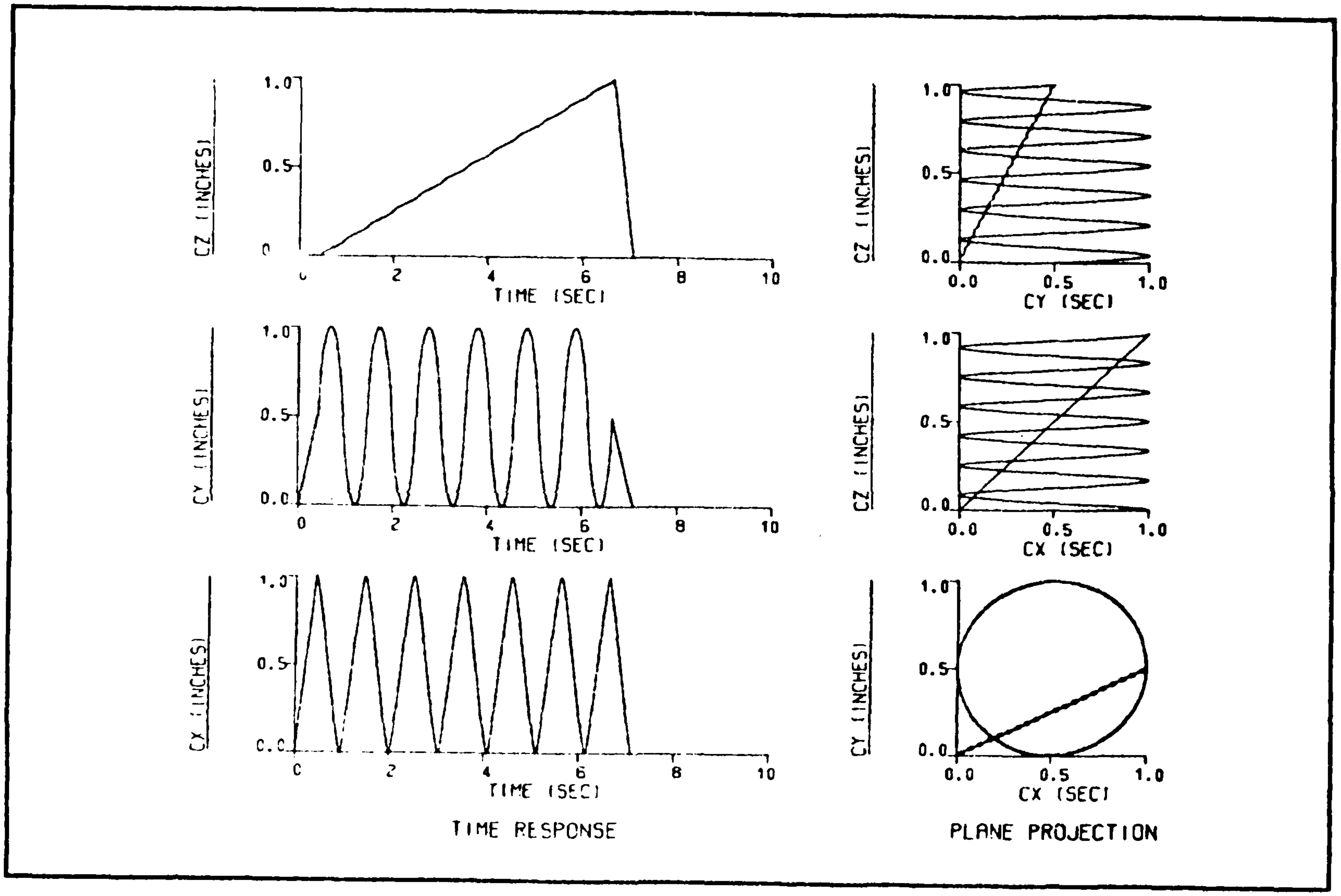


Figure 6.12 Helix Path 2, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

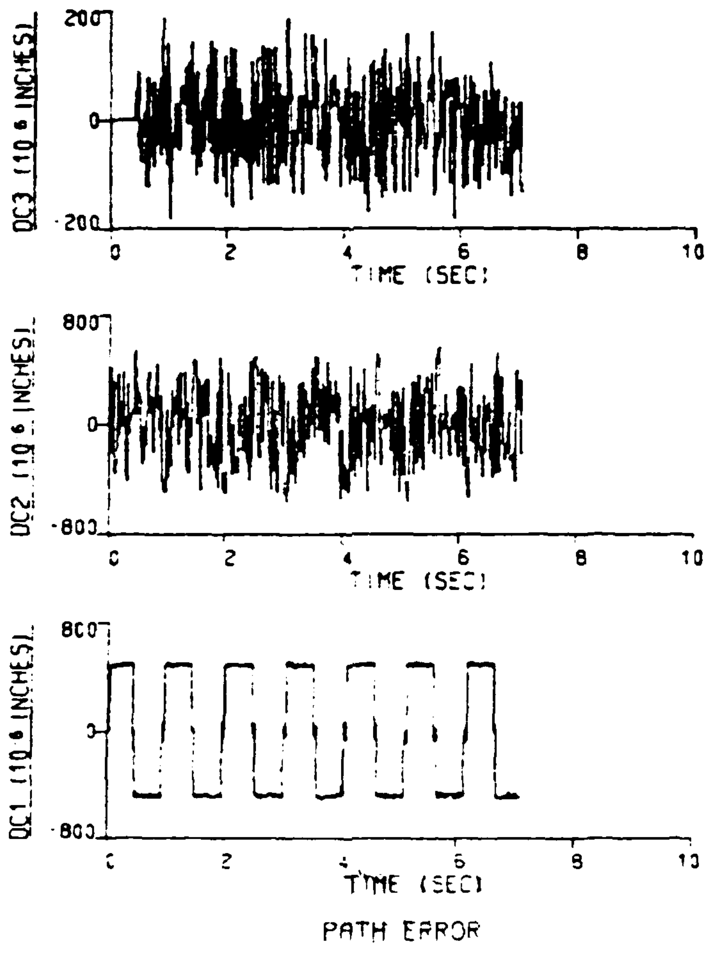


Figure 6.12(c) Helix Path 2, STEP Excitation, Position Method. X, Y, and Z Path Error.

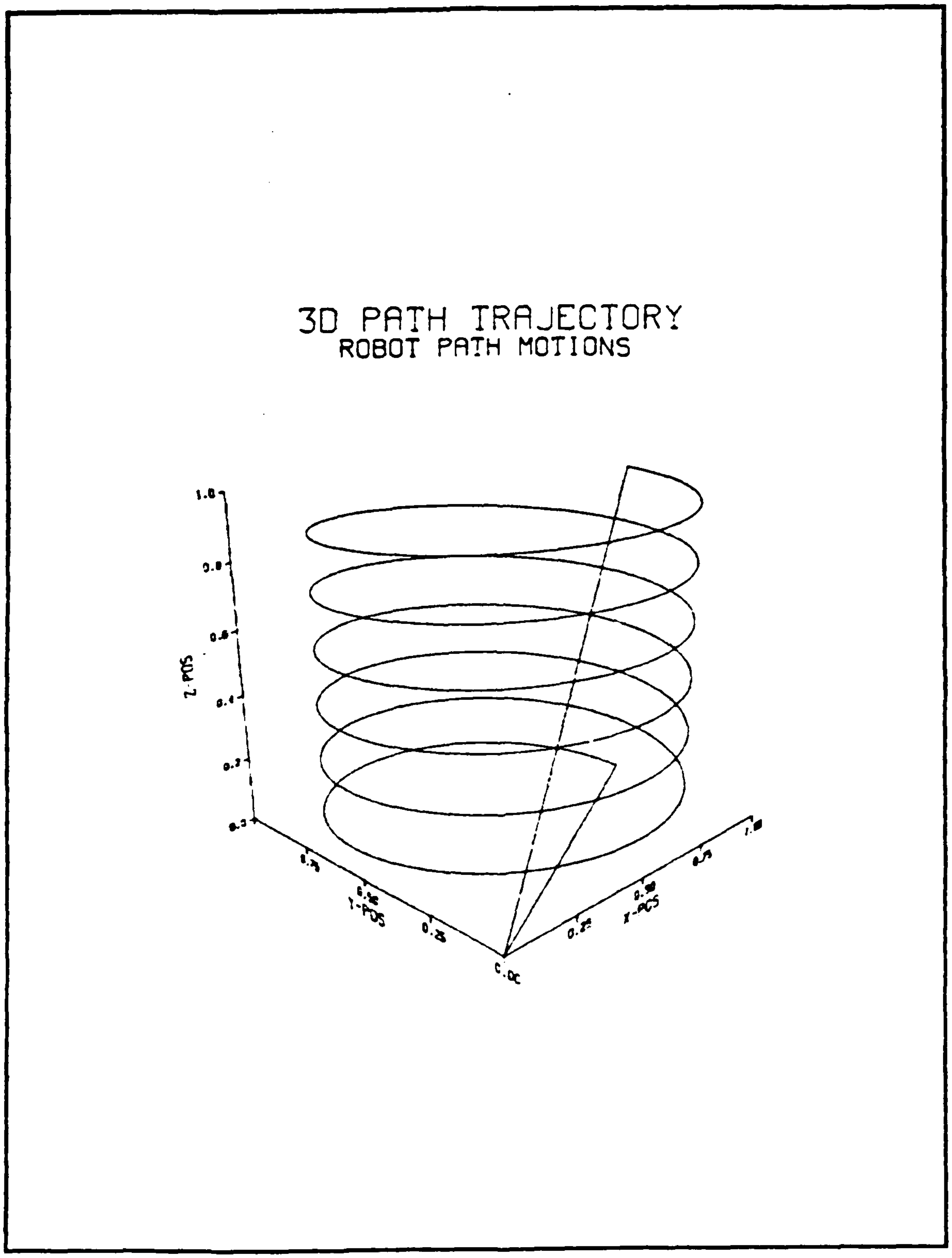


Figure 6.12(d) 3-Dimensional View of Helix Path 2.

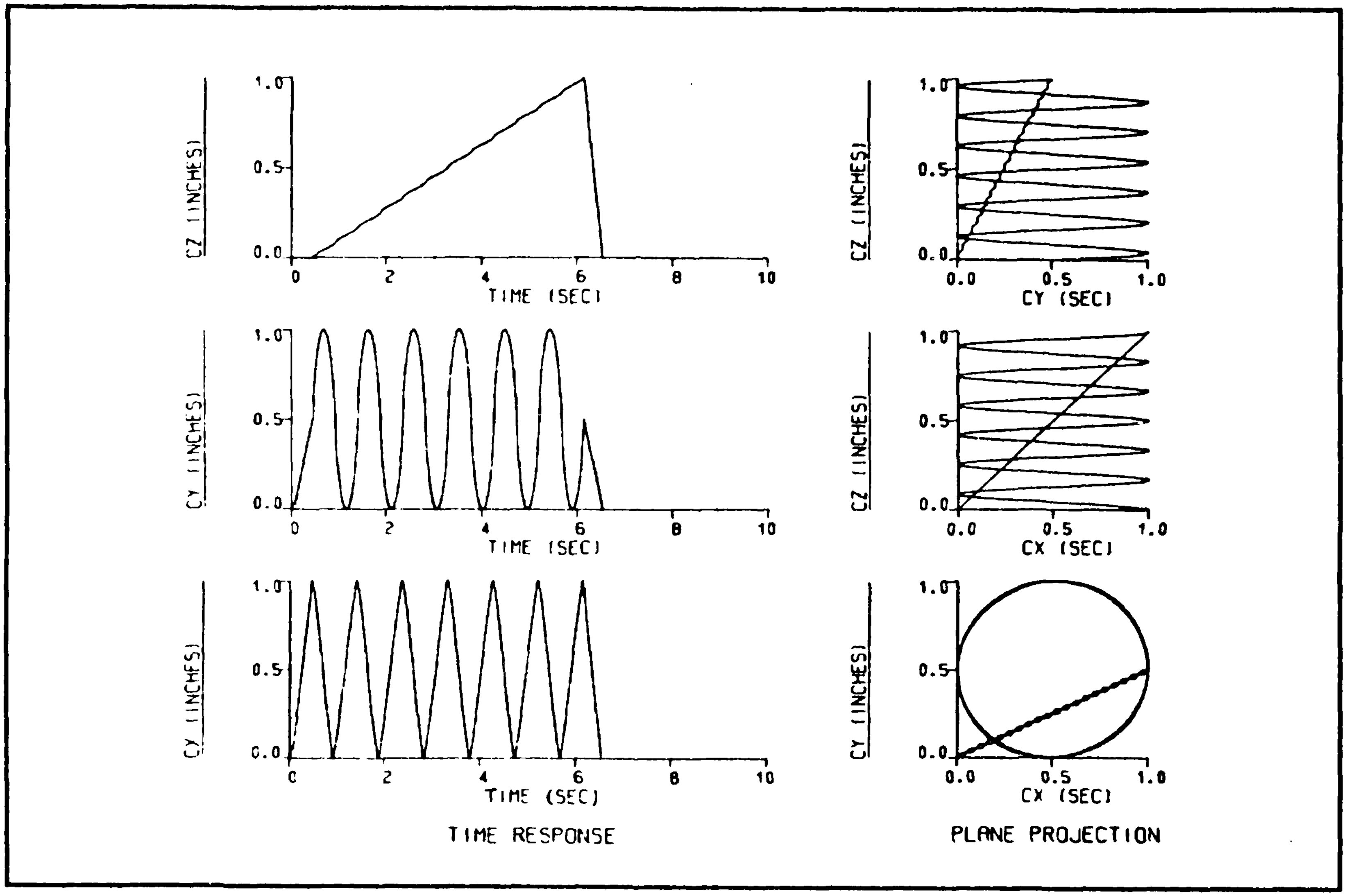


Figure 6.13 Helix Path 2, RAMP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

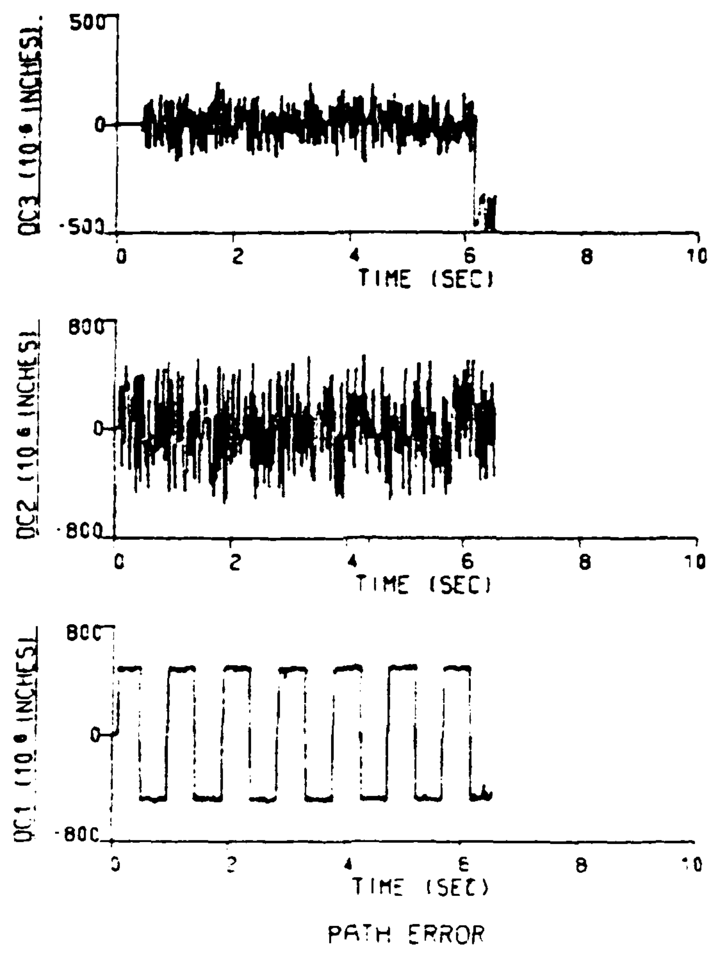


Figure 6.13(c) Helix Path 2, RAMP Excitation, Position Method. X, Y, and Z Path Error.

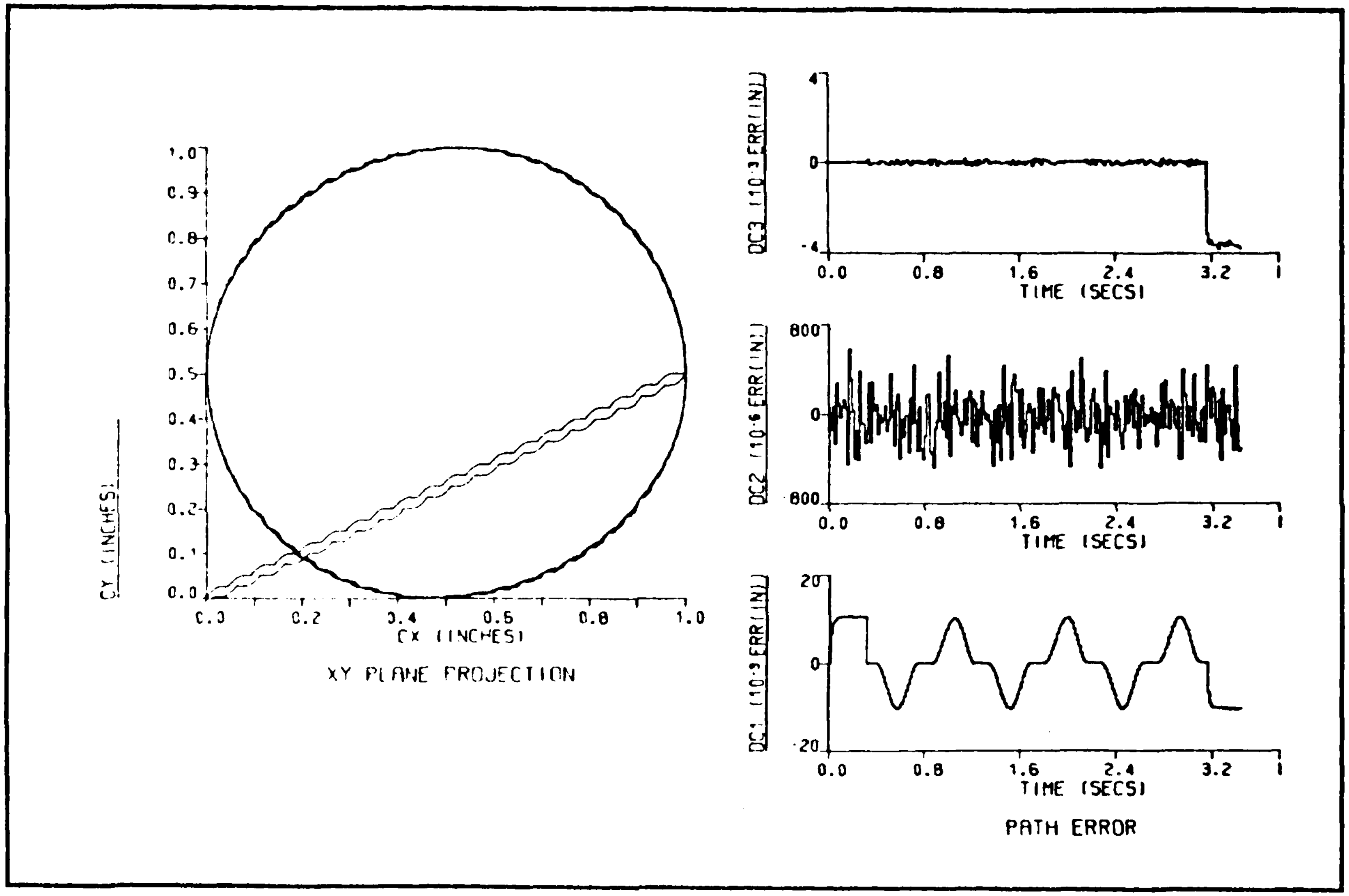


Figure 6.14 Helix Path1, STEP Excitation, Velocity Method. [T=0.015].
XY Projection and X, Y, and Z Path Error.

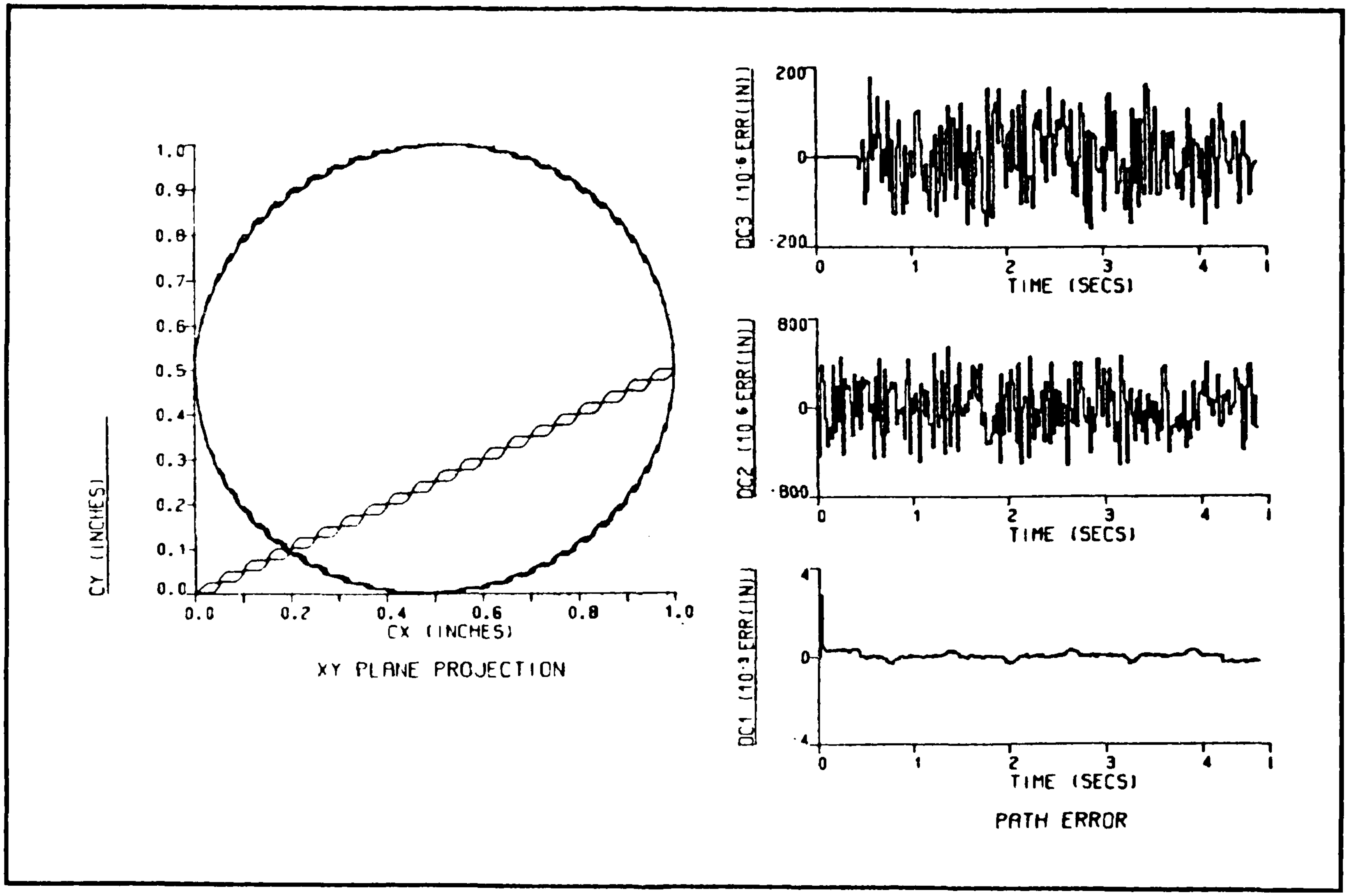


Figure 6.15(a) Helix Path 1, RAMP Excitation, Velocity Method. [T=0.020].
XY Projection and X, Y, and Z Path Error.

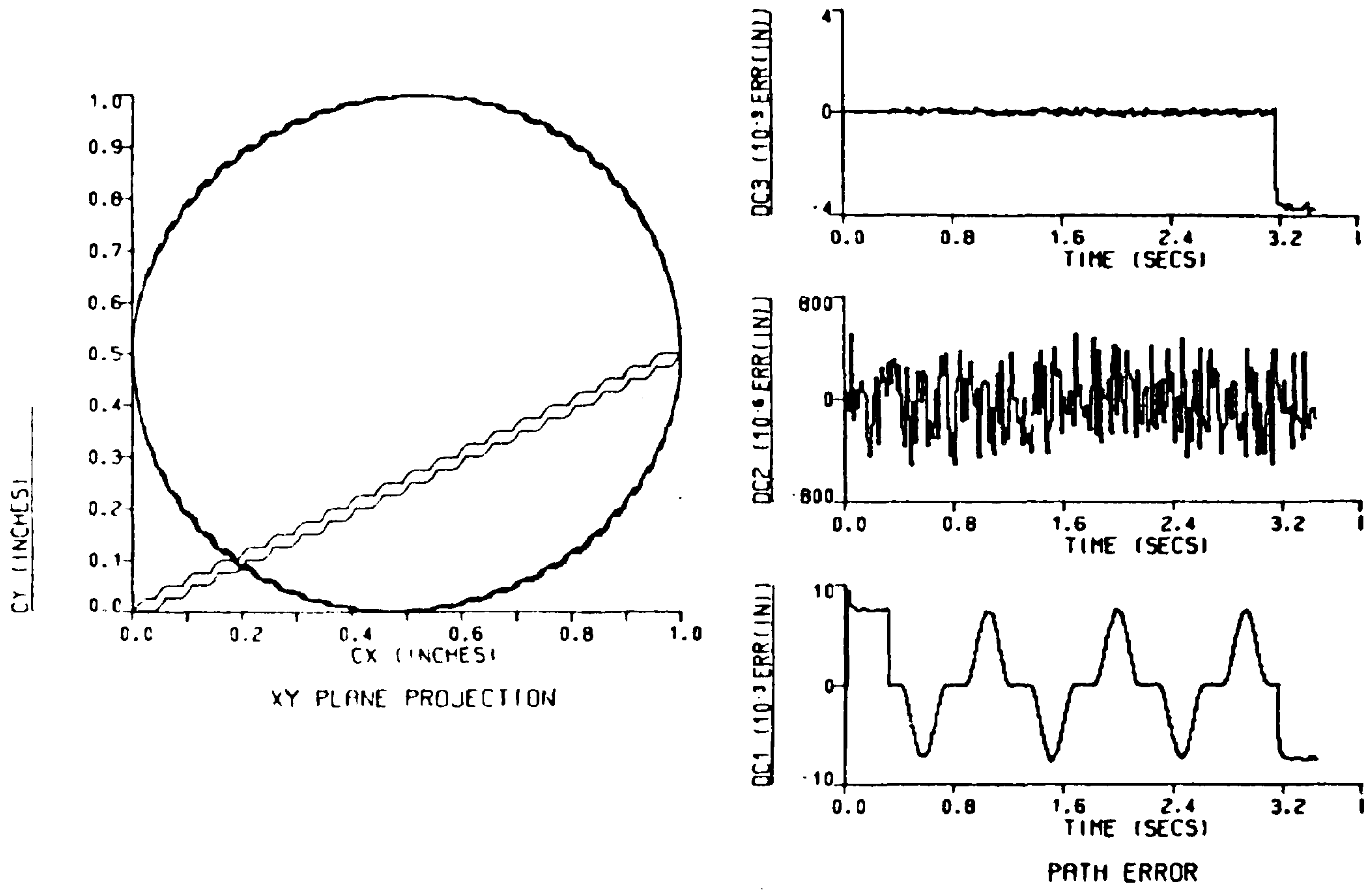


Figure 6.15(b) Helix Path 1, RAMP Excitation, Velocity Method. [T=0.015].
XY Projection and X, Y, and Z Path Error.

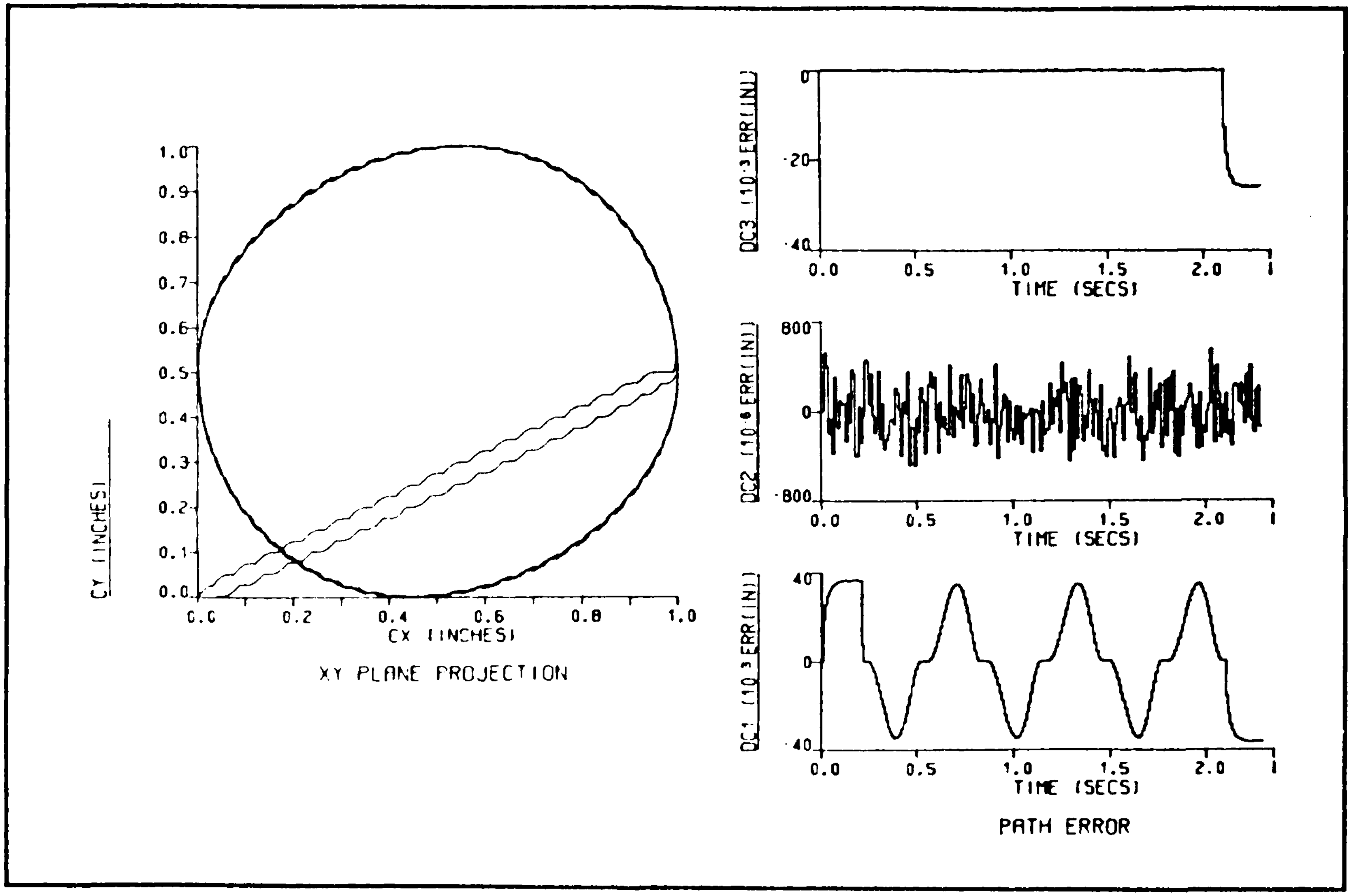


Figure 6.15(c) Helix Path 1, RAMP Excitation, Velocity Method. [T=0.010].
XY Projection and X, Y, and Z Path Error.

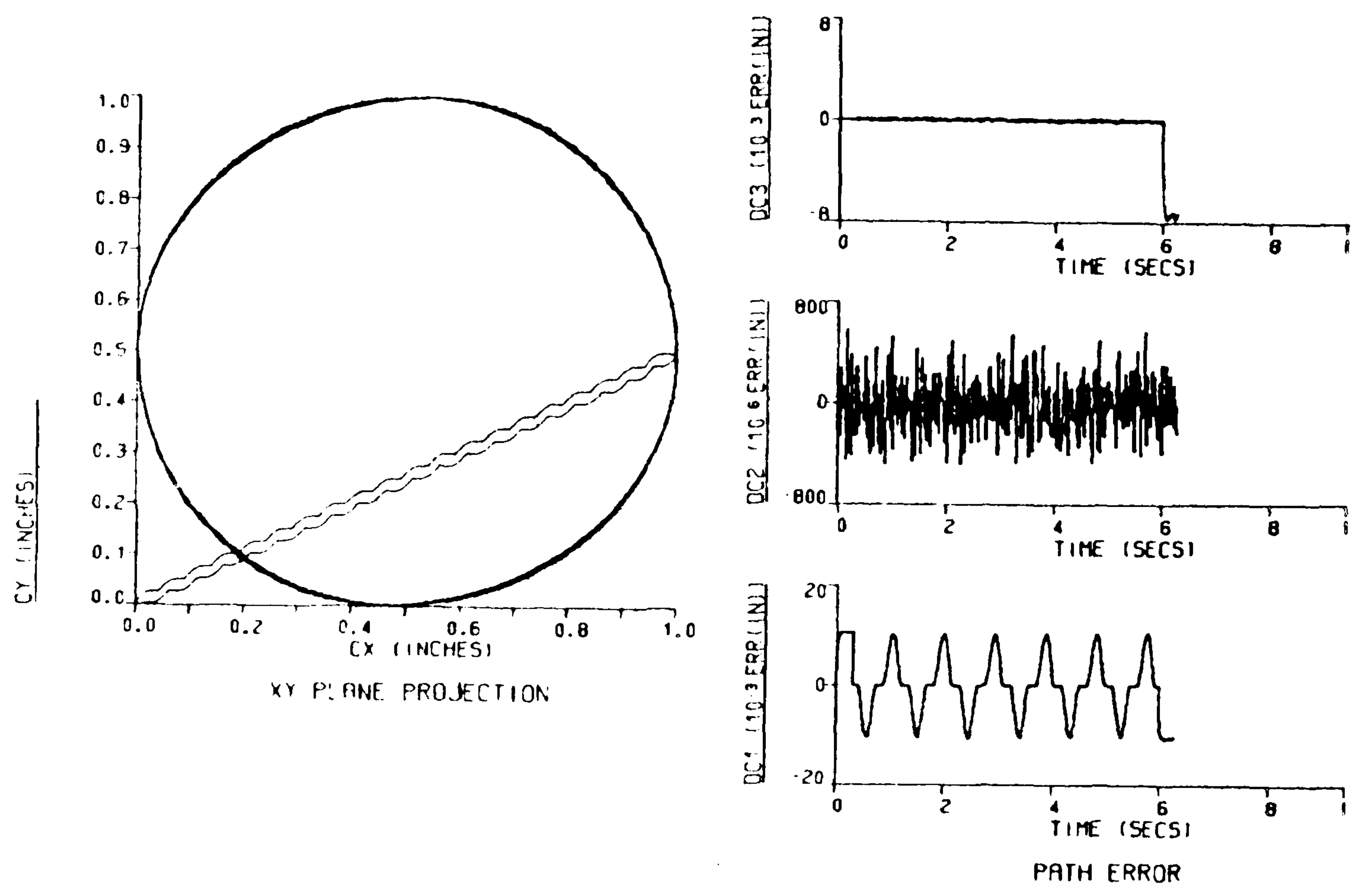


Figure 6.16 Helix Path2, STEP Excitation, Velocity Method. [T=0.015]. XY Projection and X, Y, and Z Path Error.

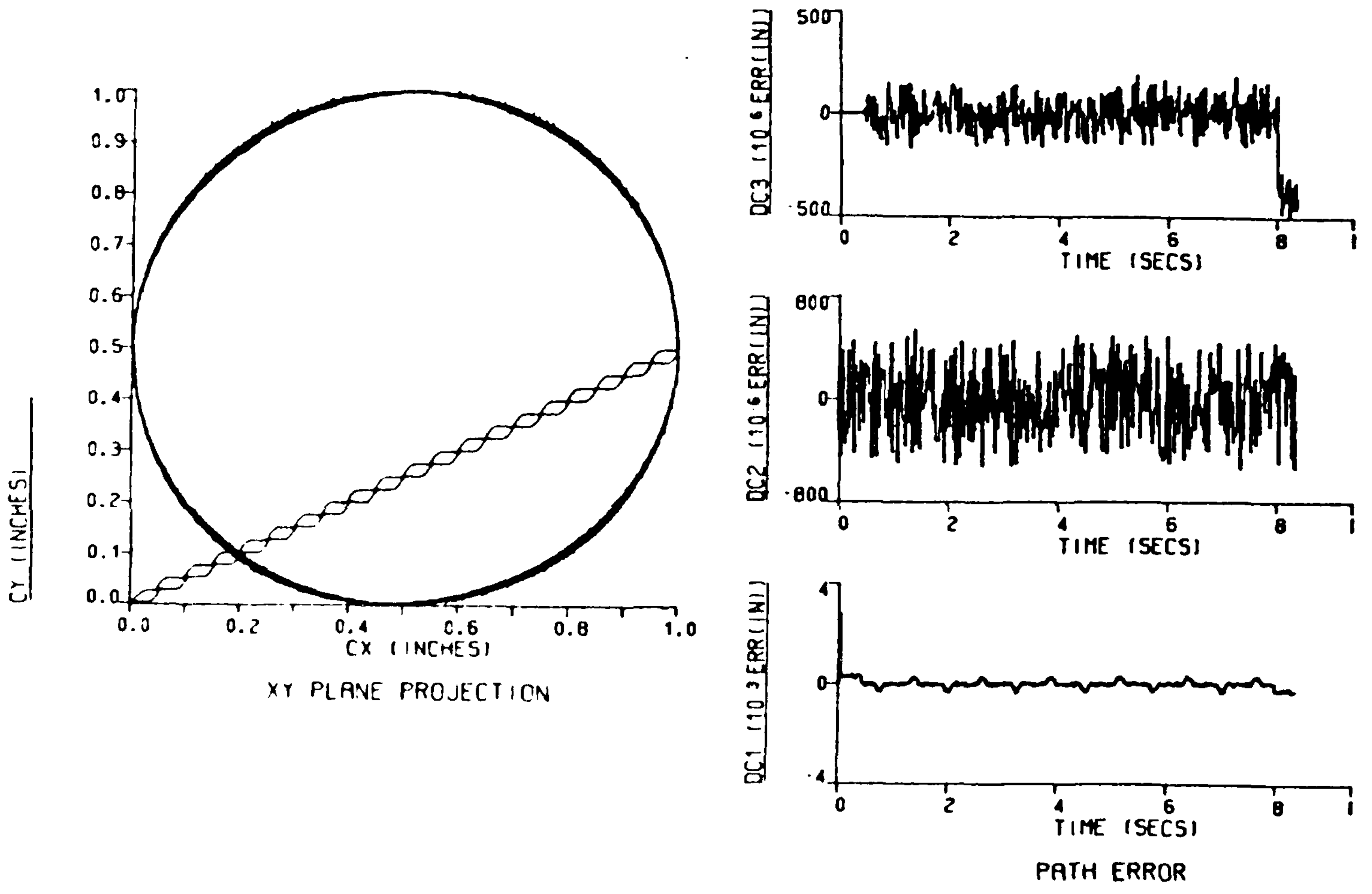
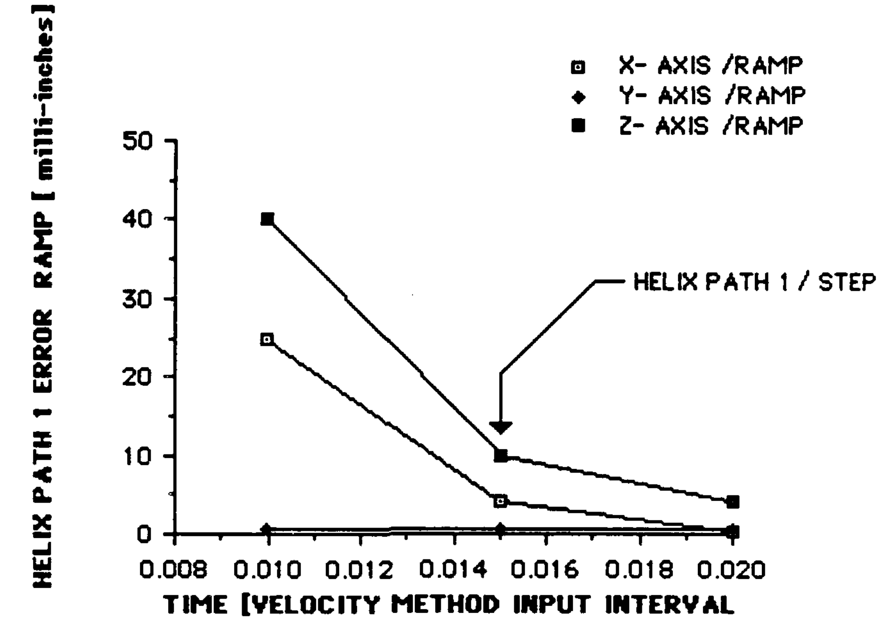


Figure 6 17 Helix Path 2, RAMP Excitation, Velocity Method. [T=0.020]. XY Projection and X, Y, and Z Path Error.



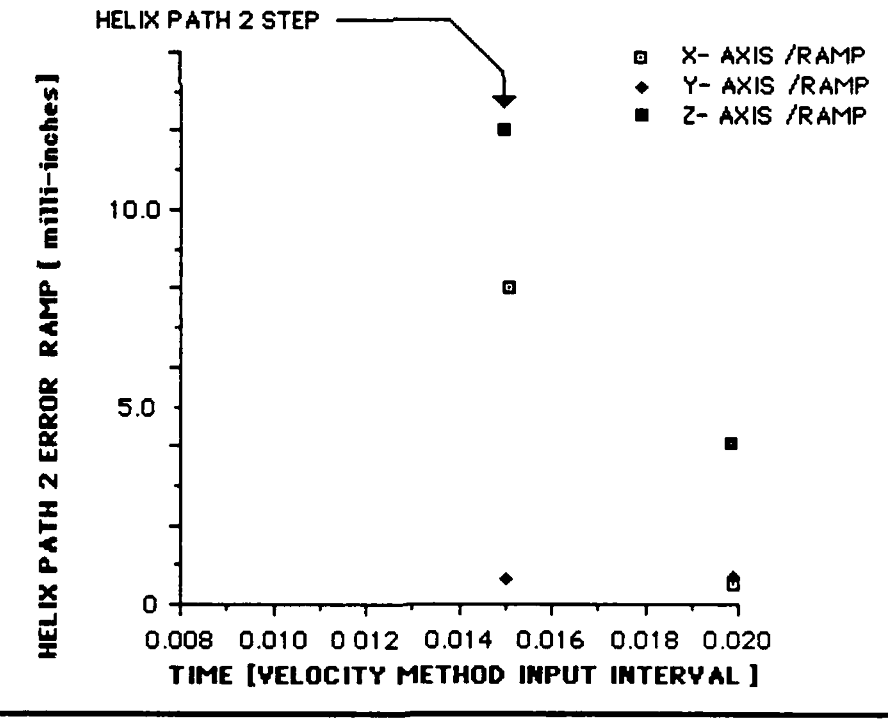


Figure 6.18 Right Circular Helix Error Comparison.

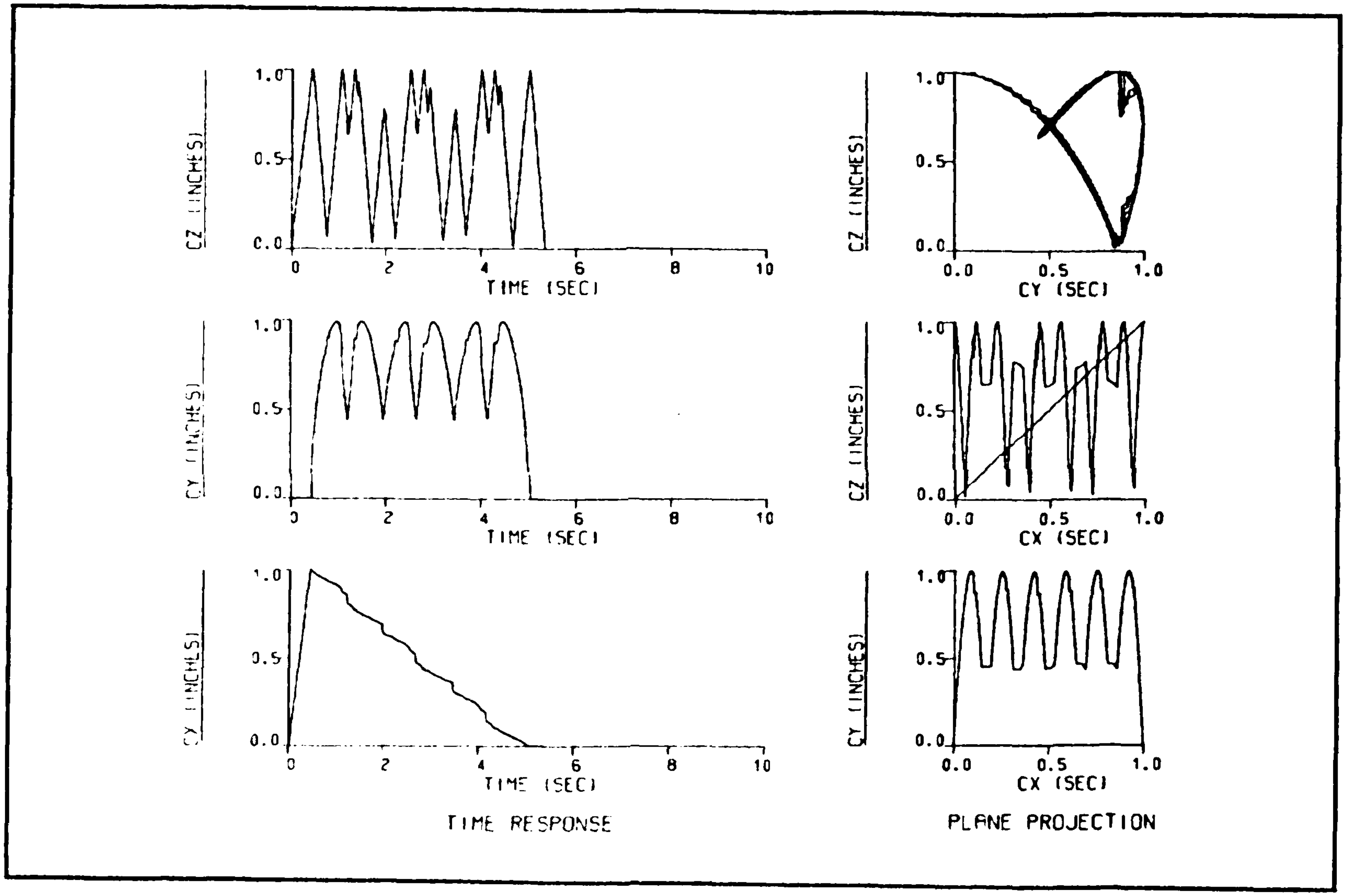


Figure 6.19 Sinusoidal Path, STEP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

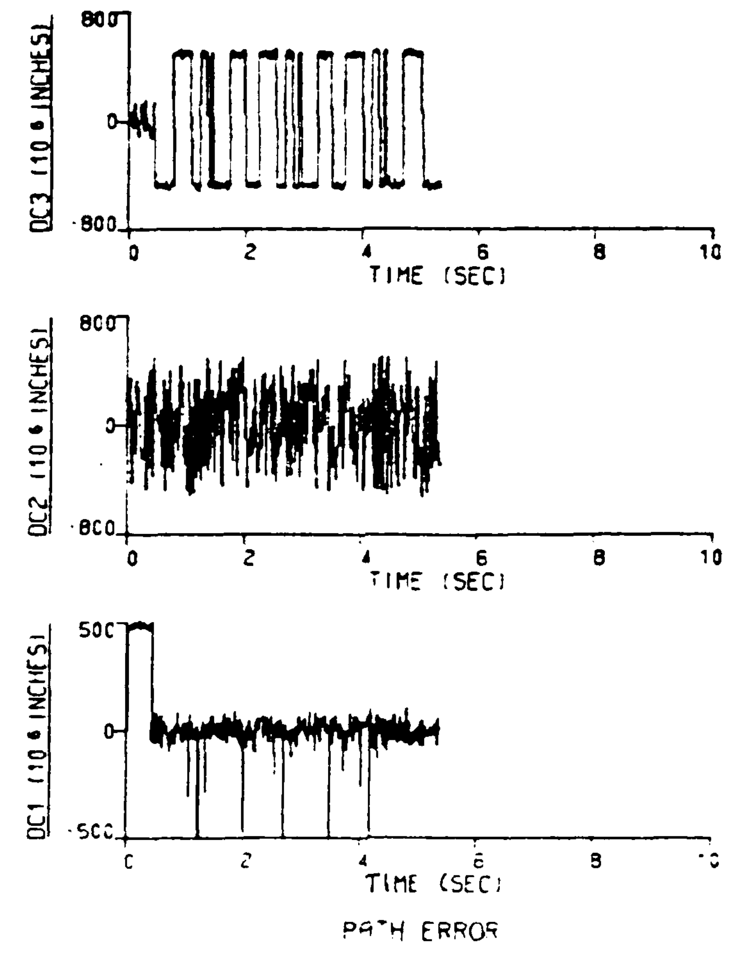


Figure 6.19(c) Sinusoidal Path, STEP Excitation, Position Method. X, Y, and Z Path Error.

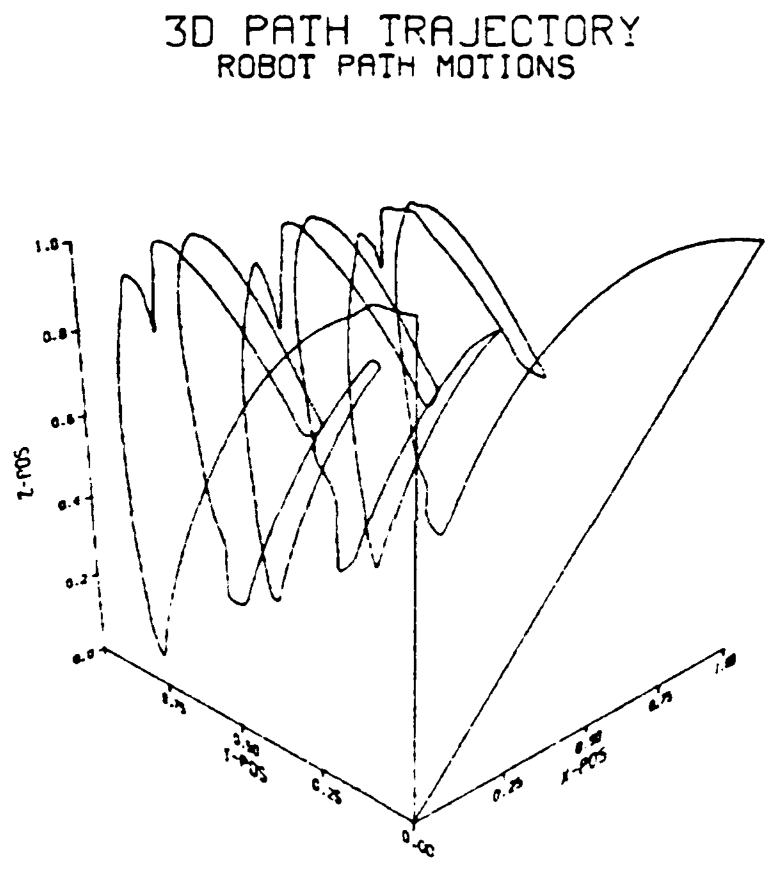


Figure 6.19(d) 3-Dimensional View of Sinusoidal Path.

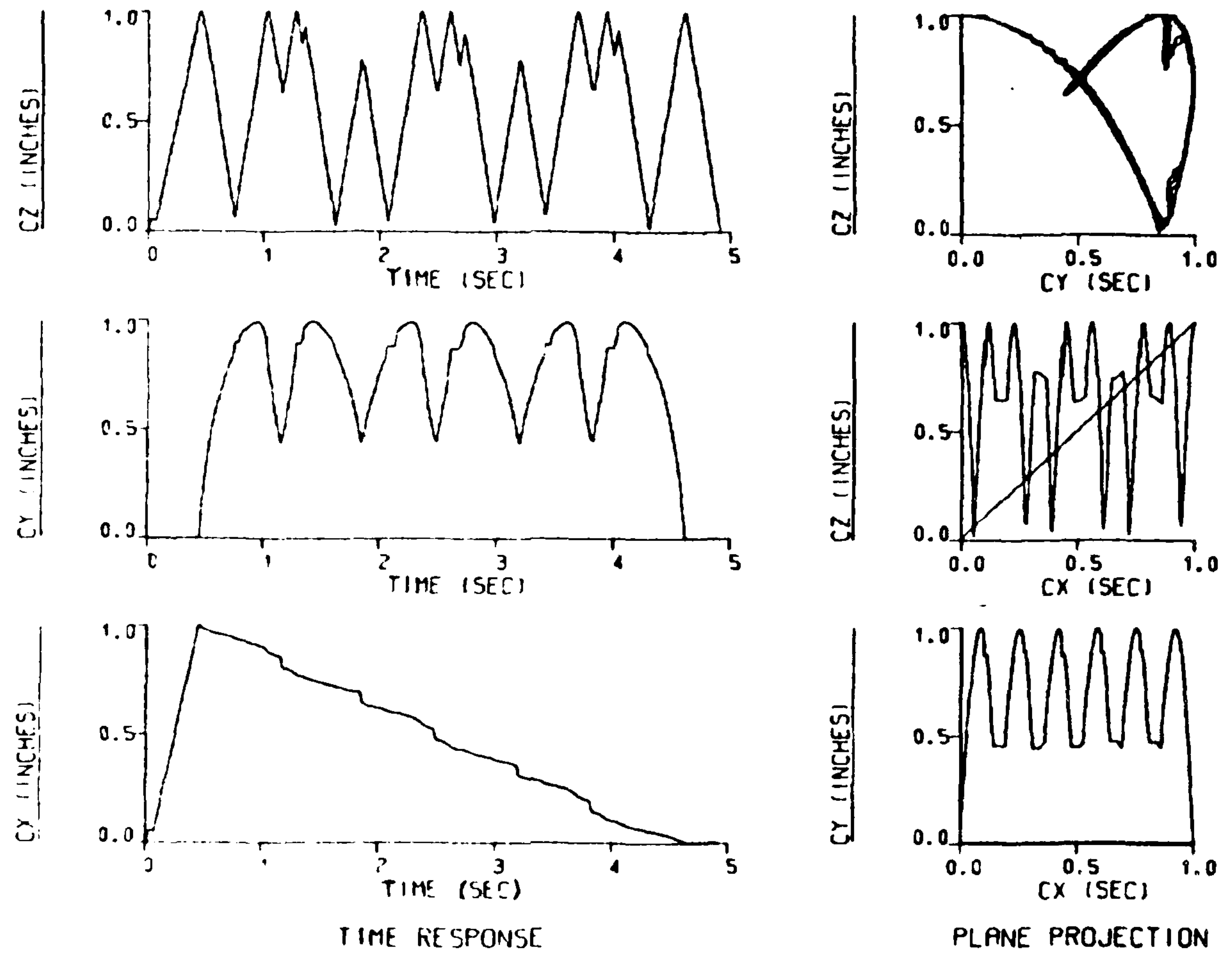


Figure 6 20 Sinusoidal Path, RAMP Excitation, Position Method.
(a) X, Y, and Z Time Response. (b) XY, XZ, and YZ Plane Projection.

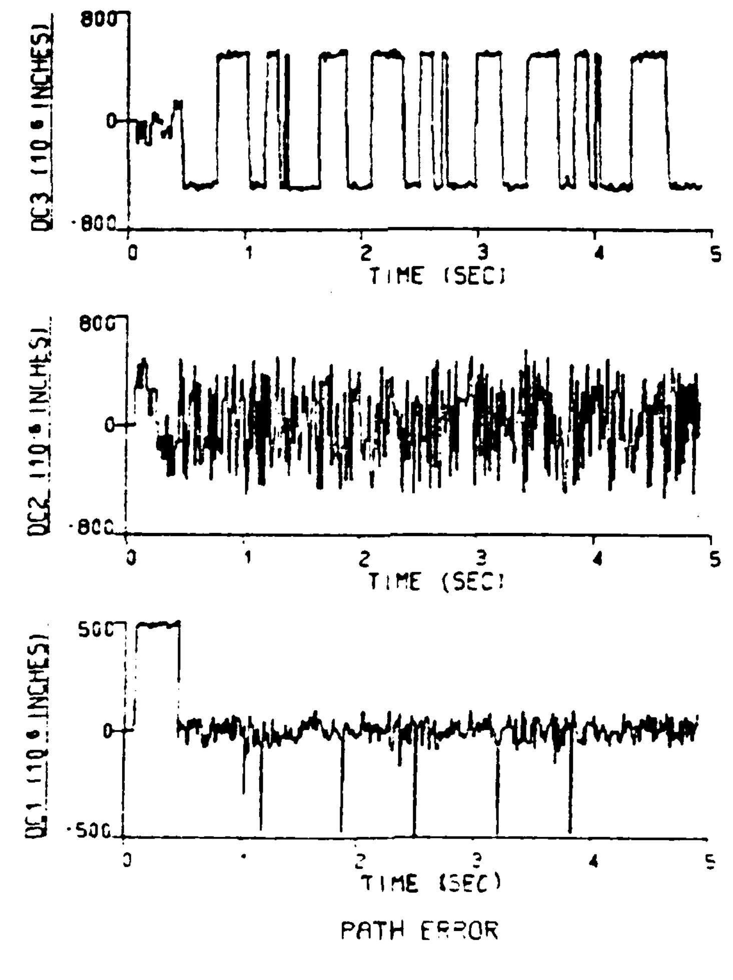


Figure 6.20(c) Sinusoidal Path, RAMP Excitation, Position Method.
X, Y, and Z Path Error.

VII. ARTICULATED ROBOT

A. INTRODUCTION

As mentioned in Chapter Two the second robot arm model that is being considered is the *articulated* (or *revolute*) robot model. Using procedures similar to that of the Cartesian Robot Arm model the lagrangian equations were solved, the computer simulation model was generated, and the DSL program was designed (Kalogiros, 1987). Appendix G shows the entire DSL program generation along with single runs that were used to verify and test the DSL program and equations. The purpose of this chapter is to tie the path generating program, written in cartesian coordinates, to the DSL program inputs, which are in articulated coordinates. To do the coordinate axis reposition an intermediate transformation to spherical coordinates was required. This also allows spherically generated path data to be converted directly into articulated coordinate path data. To interpret the results it is necessary to reconvert the articulated path back into spherical or cartesian coordinates to be plotted.

B. COORDINATE TRANSFORMATION'S

To transform the set of three tuples of cartesian coordinate points to the three tuple of spherical coordinate points is fairly simple with just a few points to keep in mind. Figure 7-1 (a) shows the range of motion for a Cartesian Robot that was used in Chapter Five while Figure 7-1 (b) shows the range of motion for the articulated robot. Each of the figures show the designed home (reference) position used and Figure 7-2 shows the regions superimposed upon each other.

The equations used to transform the data points from cartesian to spherical coordinates are:

$$\rho = ((XP-.5)^2 + (YP-.5)^2 + (ZP-.5)^2)^{1/2} \qquad \text{[eqn 7-1]}$$

$$\theta = \arccos [((b)^2+(c)^2-(a)^2)/(2bc)] \qquad \text{[eqn 7-2]}$$

$$\varphi = \arccos [((d)^2+(\rho)^2-(f)^2)/(2d\,\rho)] \qquad \text{[eqn 7-3]}$$

where

ρ is the radial unit length from (.5,.5,.5) to the point being transformed (XP,YP,ZP).

θ is the angle parallel to the XY plane referenced in the x direction to (XP,YP,ZP).

φ is the angle of ρ, referenced in the z direction through the point (.5,.5,.5).

a,b,c,d, and f are lengths described in Figure 7.1 (b).

Equation 7-1 is the three dimensional Pythagorean principle, and equations 7-2 and 7-3 relate the coordinates by use of the Law of Cosines [Appendix A].

Once in spherical coordinates the three tuple must be transformed into something useful to the robot arm. Called the articulated coordinate system, it consists of a three tuple of angles referenced to a spherical coordinate ρ maximum of .5 units. Figure 7.3 shows the spherical to articulated reference system whose resulting equations are:

$$CP1 = \theta \qquad [eqn\ 7\text{-}4]$$

$$CP2 = \pi - \varphi + \arccos(2\rho) \qquad [eqn\ 7\text{-}5]$$

$$CP3 = \pi - 2(\arccos[2\rho]) \qquad [eqn\ 7\text{-}6]$$

Several assumptions were made in the calculations:

1. Both of the articulated robot arms are of the same length. If they are not then the program will have to be adjusted by using the equations of Figure 7.3.
2. The robot operates in the 'up' elbow mode. CP1 and CP3 are the same in either mode. CP2 has a change in the sign of the arccos term [+ for 'up' elbow and - for 'down' elbow]. Figure 7.3 also demonstrates the 'down' elbow configuration, however all of the programs are solely 'up' elbow. 'Up' and 'down' elbow considerations are important when considering loading and obstacle avoidance.
3. To make the programming simpie it was assumed that the spherical coordinate system exists within a 1 unit diameter sphere giving a maximum radius of .5 units and arm lengths of .25 units. The articulated robot coordinate system really doesn't consider actual arm lengths in the movements, however calculations from spherical to articulated must have magnitudes less than or equal to 1 for proper operation of the cosine and sine functions. Also any scaling or limiting done to the cartesian and spherical coordinate systems prior

to converting to the articulated coordinate system must be undone or at least considered when interpreting the results.

To interpret the motion that the robot is making it is necessary to convert the articulated coordinate system back into either the spherical or cartesian coordinate system. Figure 7.4 contains a listing of the available coordinate conversion routines as well as a listing of the order in which to program them. The program TRANSFOR.FORTRAN when set up correctly allows transformation from spherical to articulated and then back to cartesian and articulated to spherical. The reason for transforming the first series back to spherical is to check that the final product is the same as the original if no scaling or limiting is done and it also gives the desired simulation results when scaling or limiting is used prior to the simulation.

C. SCALING AND LIMITING ROUTINES

To insure that the motion of the robot is within the range of possible movement of the robot arm it is necessary to scale or limit the motion. Scaling takes the largest value of distance and normalizes the path three tuples to one. There is also a scaling factor [SF] which then scales the result to the desired values. Scaling is available for the X, Y, and Z values of the cartesian coordinate system or for ρ in the spherical coordinate system. It is important to remember that if the SF is between 0 and 1 then the path shrinks and the maximum value of the desired path will be less than or equal to one while if the SF is greater than one then the path will be greater than 1. To insure proper operation when converting to the articulated system it is mandatory that the spherical system be scaled (if desired) and limited.

Limiting takes the results of the inputted path data, finds all the values greater than 1 in the desired cartesian path [.5 for ρ in the spherical] and replaces these values with 1. This prevents running the robot arms past their capabilities and with scaling can lead to some interesting paths. Both scaling and limiting routines are listed in Figure 7.4 and Appendix G.

D. TEST RUN RESULTS

To test the program for proper operation of the DSL program the paths generated for the cartesian robot were converted and used as outlined in Figure 7.5. An adjustment was made to the DSL program which inserted the first three tuple as the 'home' position so that the paths would not have to be reconstructed using the spherical home as the starting point. This reduces the errors in the first step to within an acceptable amount in both time and distance and does not have any effect if you start from 'home' [the first position should be home in any case]. Table 7.1 show the paths that were used and the Figures associated. When viewing the three dimensional plots there is a line from the last point to zero that is not resident in the cartesian data, either after conversion, bypassing simulation (SCPATH . DATA) , or after simulation and conversion back into cartesian coordinates (DCPATH . DATA), that is unaccounted for and is an error in the DISSPLA System program. All of the runs are in the STEP Excitation, Position Method format. Programs for the RAMP Excitation, Position Method and the STEP and RAMP Excitations, Velocity Method were written and modifications are located in Appendix D, but no runs were done to check for proper operation because of the time involved in even the shortest run. Twenty to thirty CPU minutes are not uncommon when calculating a 500 point path with the current simulation model.

TABLE 7.1
ARTICULATED ROBOT ARM PATH MOVEMENT

PATH PROGRAM	TYPE PATH	FIGURES	NUMBER OF POINTS
PATH2.data	LINEAR 1	7.5 (a)-(e)	80
PATH3.data	LINEAR 2	7.6 (a)-(e)	139
PATH4.data	CIRCULAR	7.7 (a)-(e)	505
PATH5.data	HELICAL 1	7.8 (a)-(e)	231
PATH5A.data	HELICAL 2	7.9 (a)-(e)	420
PATH6.data	LINEAR	7.10 (a)-(e)	233
PATH7.data	SINUSOIDAL	7.11 (a)-(e)	345

(a) TIME RESPONSE (radians)
(b) PATH ERROR (milli-rads)
(c) ORIGINAL PATH IN 3-DIMENSIONS
(d) DIRECT CONVERSION PATH IN 3-DIMENSIONS
(e) SIMULATED CONVERSION PATH IN 3-DIMENSIONS

As with the Cartesian Simulation model for Position Method the accuracy was determined by E [VALMIN] dropping below the value of 0.001 inches. Path Error is calculated by taking the difference between the desired position and the computer simulation model output and E is calculated by taking the difference between the desired position and the motor simulation model output.

During the testing of the Articulated Model some interesting problems were encountered. Due to using paths designed in cartesian coordinates, the articulated coordinate inputs to the simulation are not controlled as to the size of radian step from one point to the next. When far from the center of the Articulated Coordinate System there is not any problem as the radian step is small but as you get closer to the center of the coordinate system a small step in Cartesian Coordinates result in a large step in Articulated Coordinates. Consider, for example, a step from one side of the Articulated Coordinate System origin directly through the center (0.5, 0.5, 0.5) to the other side of the origin is translated into a 180 degree or π radian step in the Articulated Coordinate System. Associated with this problem is that with the simulation we can assume an infinitely thin, essentially nonexistent arm, where in reality there is a lot of space where the arm can not go due to support structures. With this in mind it makes no sense to worry about going through the Articulated origin because the space is already occupied. "Home" has also been described as (0.5, 0.5, 0.5) and any time that it is hit from whatever direction the Arm whips around to "home" in the articulated coordinate system (0.0, 3π/2,0.0) and then back to the next position. By devising a scheme to prevent the arm from entering a sphere around the articulated origin most of the problems can be avoided. This was not implemented as none of the paths went close to the origin.

The next problem came about in the definition of CP1, which is the first axis in the Articulated Coordinate System. Measured in radians the arm travels from 0.0 to 2π radians without any difficult. When going counter clockwise from a number near 2π to a number just above zero a problem occurs. With the type of control we are using the Arm would try to respond by going clockwise back around the axis center until it arrives at the new number. This is undesirable and occurs in most of the paths that were generated. To solve the problem a comparison of the next position with the old position is done. If it meets the criteria described above then the old position is subtracted from 2π and renamed the old position. This is easy to implement in software but in hardware may be difficult.

All of the Path errors are within the desired VALMIN. When looking at some of the Path Error versus time remember that 2π radians and zero are the same accounting for some of the seemingly large errors. This problem could have been remedied by taking the 2π correction mentioned earlier and applied it to both the motor output (C1) and the computer simulation output (CX) but was actually only applied to the motor output (C1). The overall results are accurate and within the desired specifications.

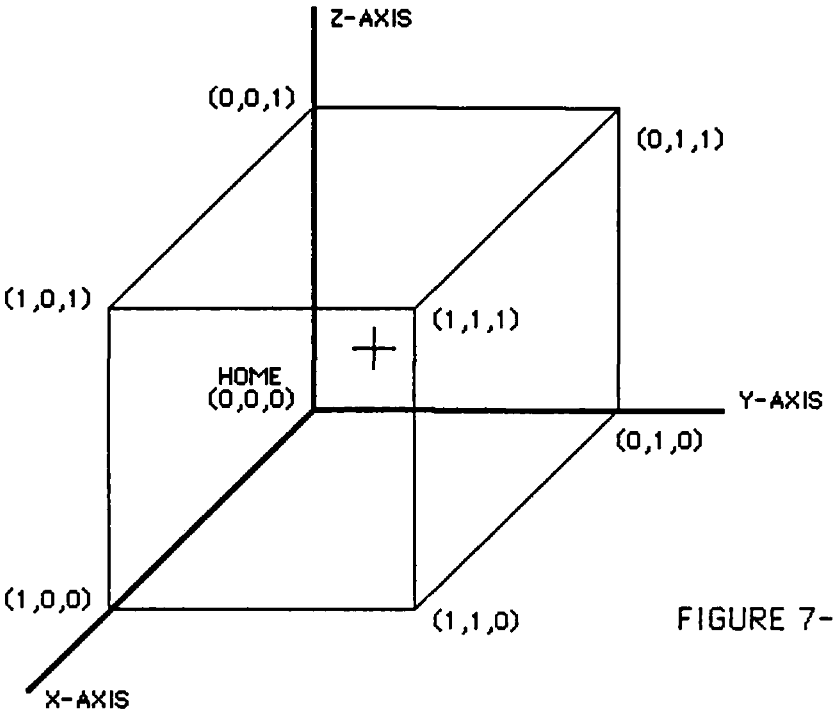


FIGURE 7-1 (a)

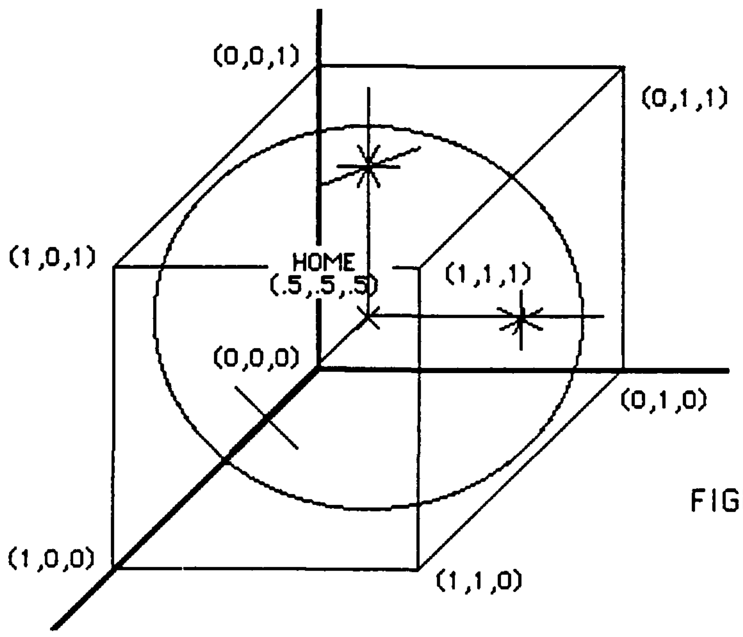
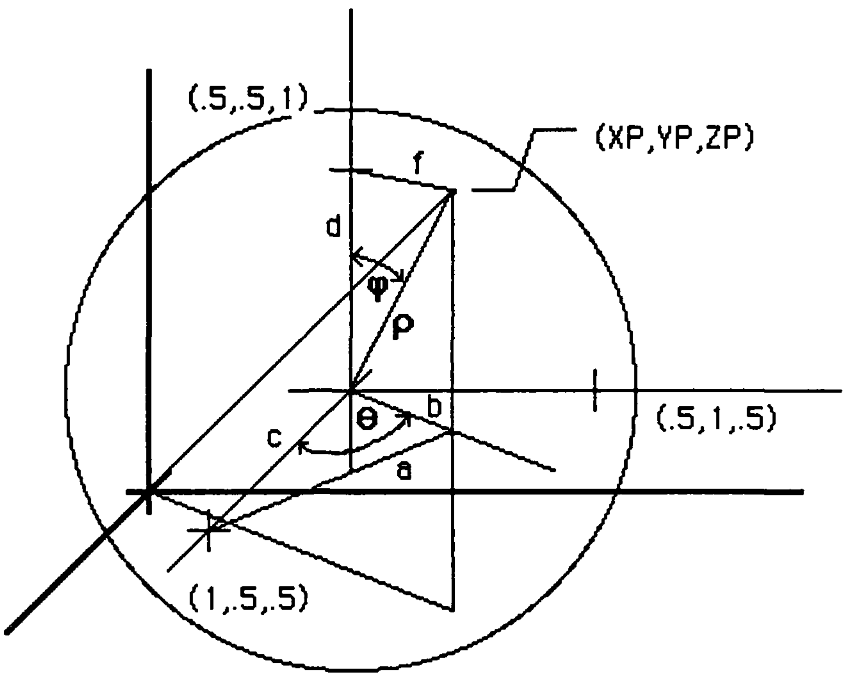


FIGURE 7.1 (b):

Figure 7.1 (a) : Cartesian Range of Motion
(b) : Spherical/ Articulated Range of Motion



Cartesian to Spherical Coordinate Transformation

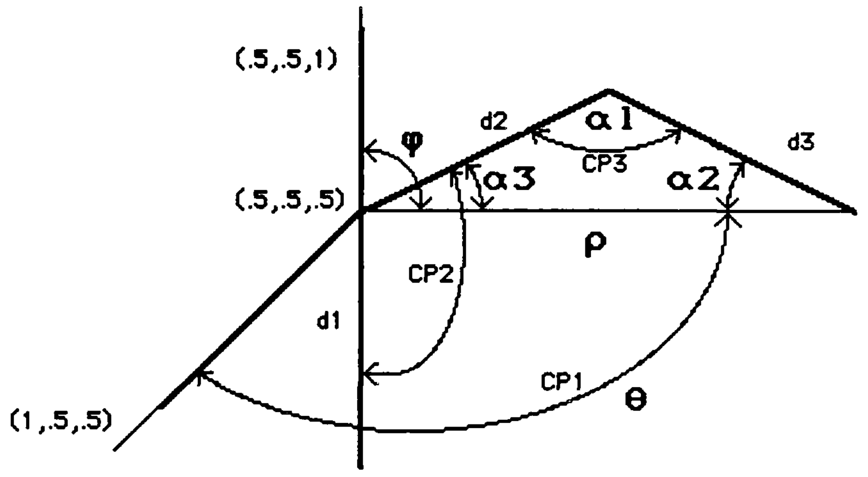
$$\rho = ((XP-.5)^2 + (YP-.5)^2 + (ZP-.5)^2)^{1/2}$$
$$\theta = \arccos[((b)^2 + (c)^2 - (a)^2)/(2bc)]$$
$$\varphi = \arccos[((d)^2 + (\rho)^2 - (f)^2)/(2d\rho)]$$

Choose

$$a = ((XP - 1)^2 + (YP - .5)^2)^{1/2}$$
$$b = ((XP - .5)^2 + (YP - .5)^2)^{1/2}$$
$$c = .5$$
$$d = .5$$
$$f = ((XP - .5)^2 + (YP - .5)^2 + (ZP - 1)^2)^{1/2}$$

Where (XP, YP, ZP) is the point being converted from Cartesian to spherical.

Figure 7.2 Cartesian to Spherical Coordinate Conversion.



Given **(ρ,θ,φ)** as Spherical Coordinates
to find (CP1, CP2, CP3) in Articulated Coordinates.

$$\begin{aligned} CP1 &= \theta \\ CP2 &= 180 - \varphi + \alpha 3 \quad \text{(elbow 'up')} \\ CP3 &= \arccos [((d2)^2 + (d3)^2 - (\rho)^2) / (2\,(d2)(d3))] \end{aligned}$$

Since $d2 = d3 = .25$ or max reach of $\rho = .5$
then $\alpha 2 = \alpha 3$ and

$$\begin{aligned} \alpha 1 &= 180 - 2(\alpha 2) = CP3 \\ &= \arccos\ [(.5 - (\rho)^2)] \end{aligned}$$

Figure 7.3 Spherical to Articulated Coordinate Transformation.

PROGRAM (FORTRAN)	PURPOSE
ROBPATH2	GENERATES LINEAR PATH 1
ROBPATH3	GENERATES LINEAR PATH 2
ROBPATH4	GENERATES CIRCULAR PATH
ROBPATH5	GENERATES RIGHT CIRCULAR HELIX 1
ROBPATH5A	GENERATES RIGHT CIRCULAR HELIX 2
ROBPATH6	GENERATES LINEAR PATH
ROBPATH7	GENERATES SINUSOIDAL PATH
TRANSFOR	
CARSP	CARTESIAN TO SPHERICAL
CARLI	CARTESIAN LIMITING
CARSC	CARTESIAN SCALING
SPHLI	SPHERICAL LIMITING
SPHSC	SPHERICAL SCALING
SPART	SPHERICAL TO ARTICULATED
ARTSP	ARTICULATED TO SPHERICAL
SPCAR	SPERICAL TO CARTESIAN
RSIMREV	ARTICULATED ROBOT MODEL SINGLE STEP INPUT
RSIMREV1	ARTICULATED ROBOT MODEL MULTIPLE STEP INPUTS
ROBCOMP1	3-DIMENSIONAL PLOTTING ROUTINE

Figure 7.4 Articulated Robot path, transformation, scaling, and limiting programs and subroutines

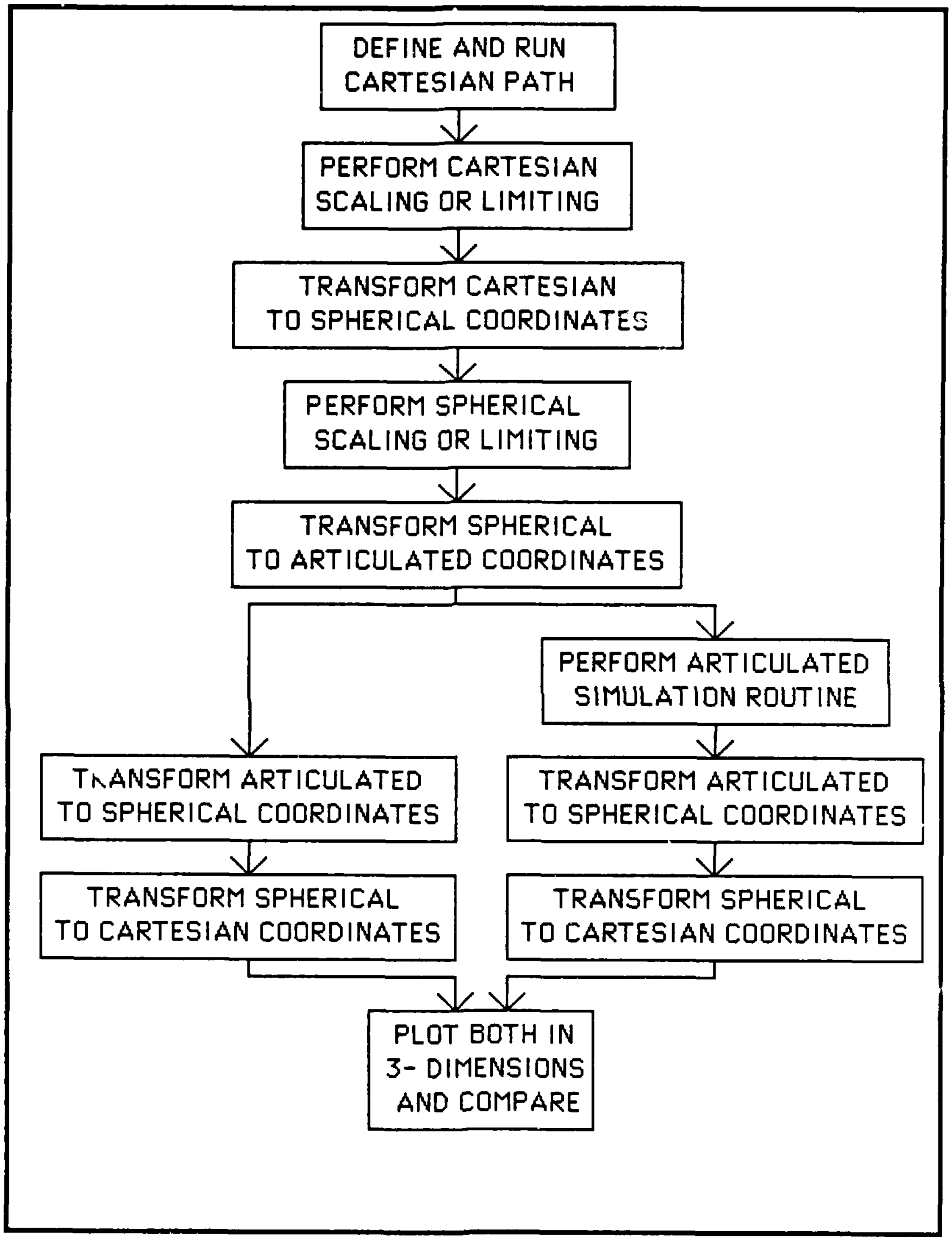

DEFINE AND RUN CARTESIAN PATH
PERFORM CARTESIAN SCALING OR LIMITING
TRANSFORM CARTESIAN TO SPHERICAL COORDINATES
PERFORM SPHERICAL SCALING OR LIMITING
TRANSFORM SPHERICAL TO ARTICULATED COORDINATES
PERFORM ARTICULATED SIMULATION ROUTINE
TRANSFORM ARTICULATED TO SPHERICAL COORDINATES
TRANSFORM ARTICULATED TO SPHERICAL COORDINATES
TRANSFORM SPHERICAL TO CARTESIAN COORDINATES
TRANSFORM SPHERICAL TO CARTESIAN COORDINATES
PLOT BOTH IN 3- DIMENSIONS AND COMPARE

Figure 7.5 Articulated Robot Path Outline.

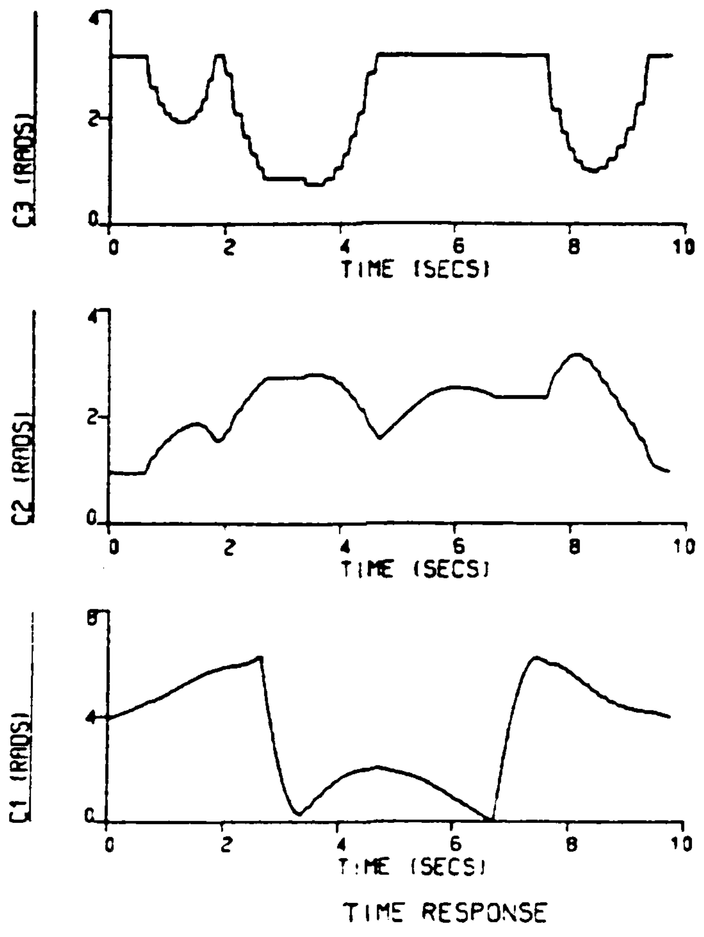


Figure 7.6(a) Articulated Robot Linear Path 1 Time Response.

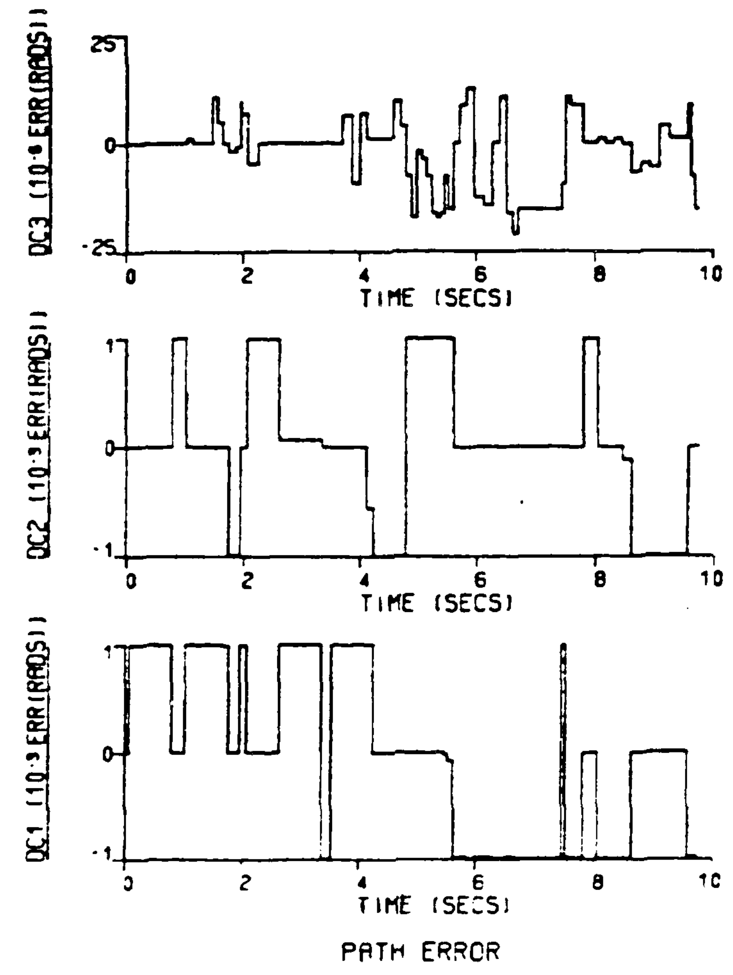


Figure 7.6(b) Articulated Robot Linear Path 1 Path Error.

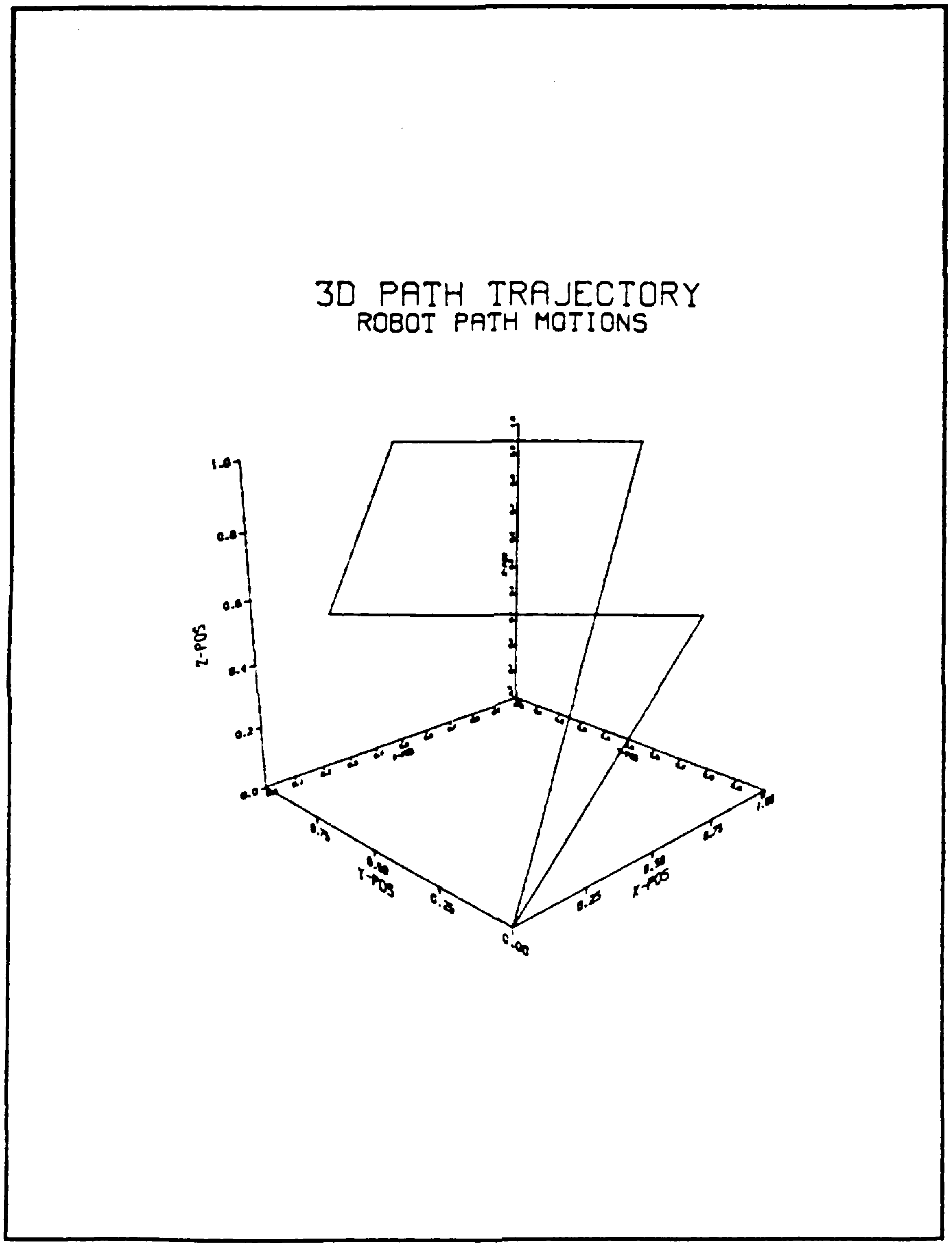


Figure 7.6(c) Articulated Robot Linear Path 1 in 3-Dimensions. [PATH2.DATA]. Pre-simulation.

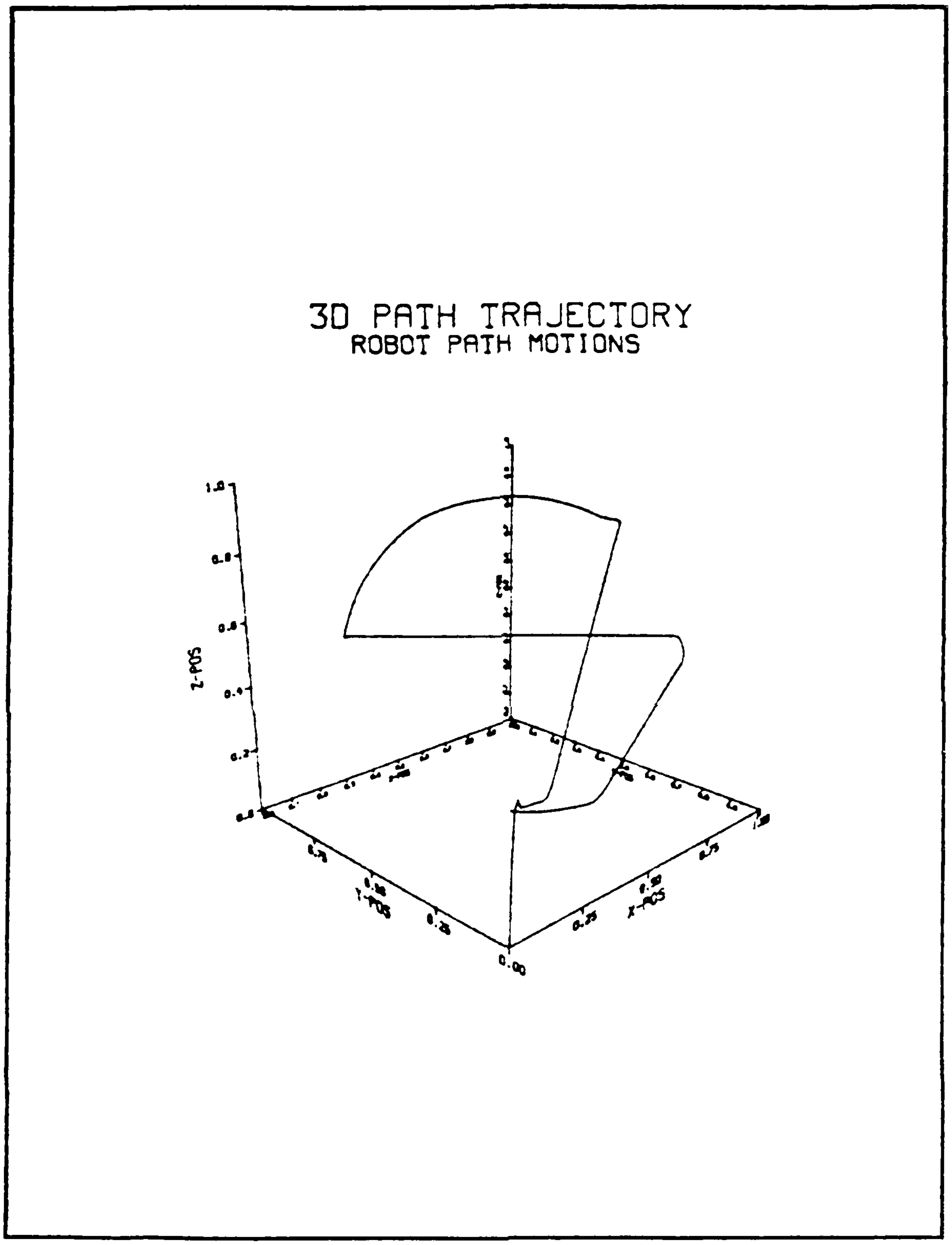


Figure 7.6(d) Articulated Robot Linear Path 1 Direct Conversion Path in 3-Dimensions. [SCPATH2.DATA]. No Simulation.

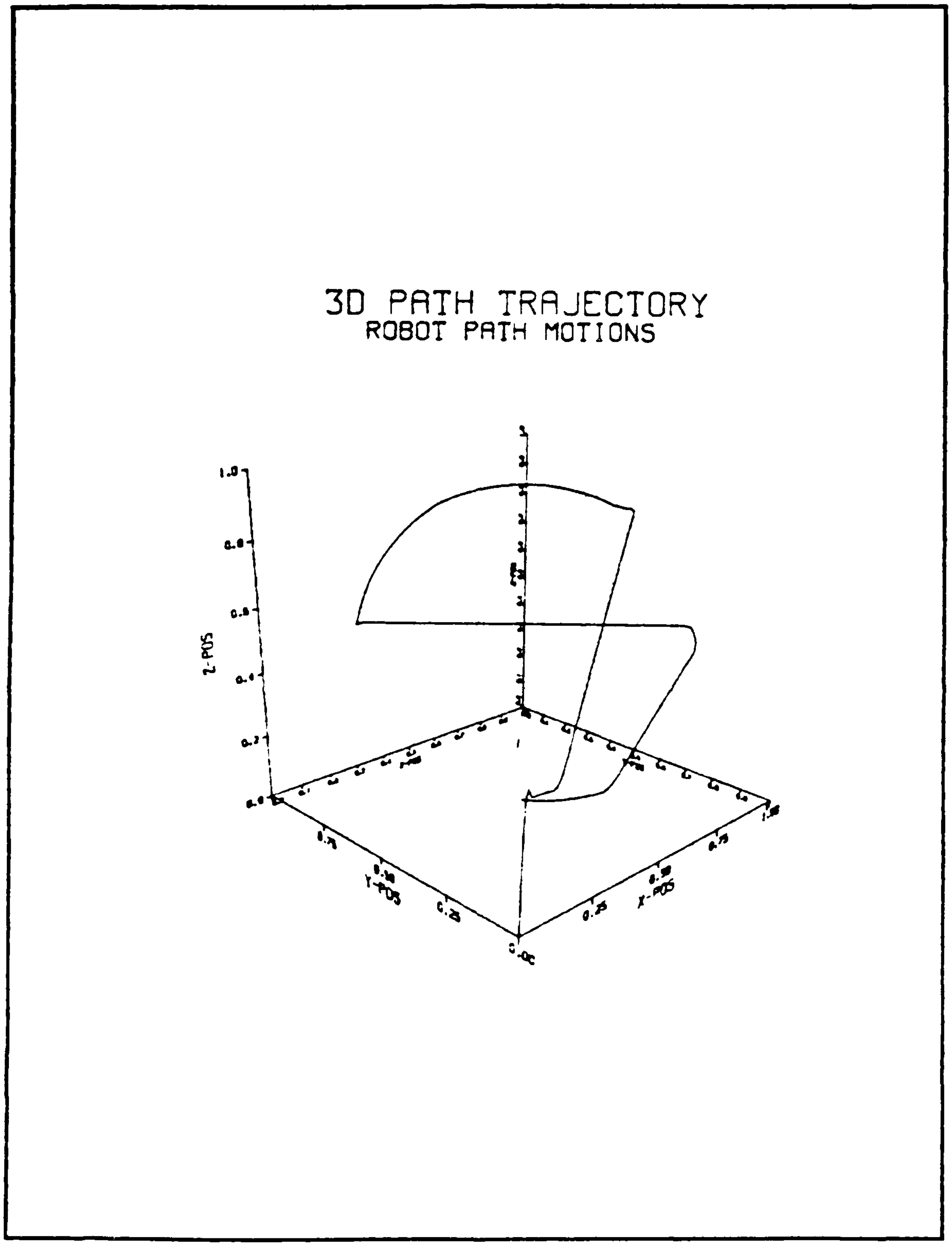


Figure 7.6(e) Articulated Robot Linear Path 1 Simulated Conversion Path in 3-Dimensions. [DCPATH2.DATA]. Post Simulation.

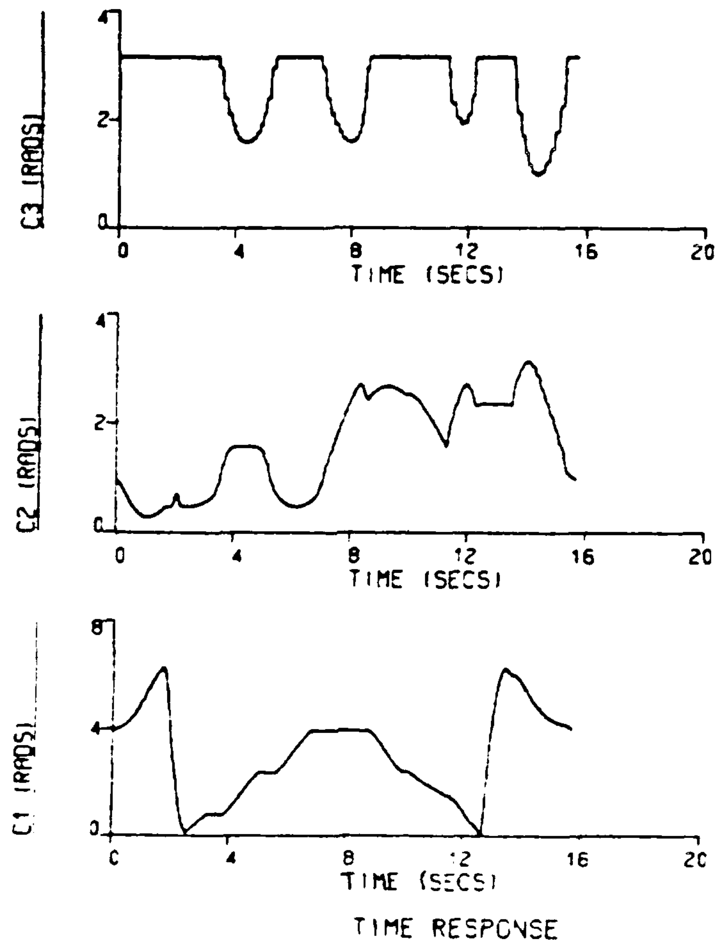
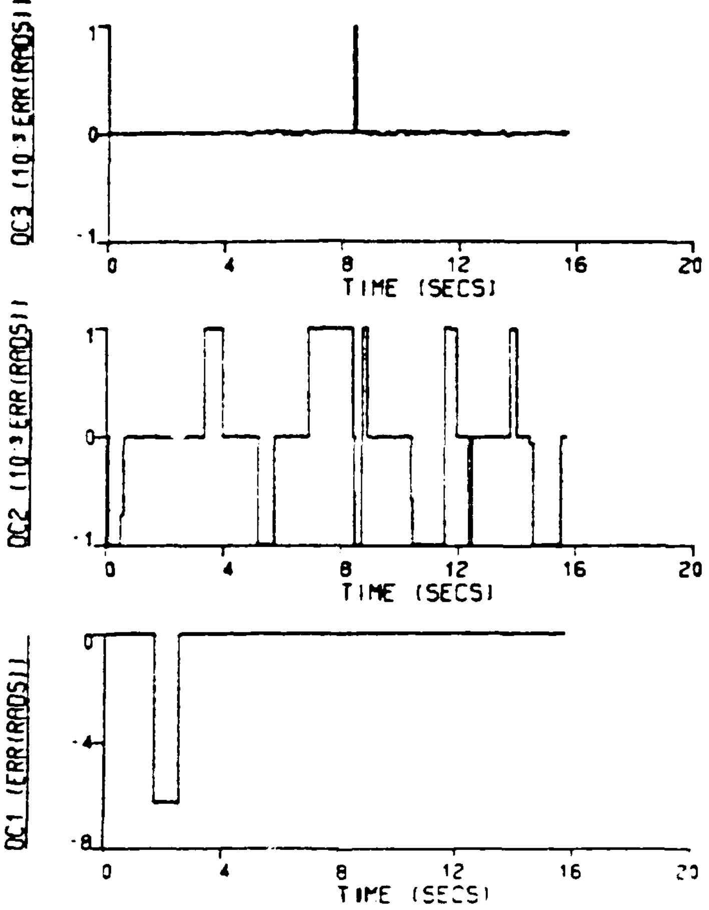


Figure 7.7(a) Articulated Robot Linear Path 2 Time Response.

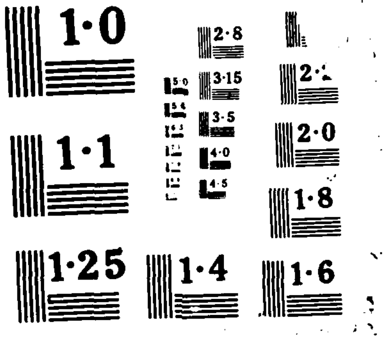


PATH ERROR

Figure 7.7(b) Articulated

AD-A190 960   PATH FOLLOWING ROBOT(U) NAVAL POSTGRADUATE SCHOOL   3/4
MONTEREY CA S G GOODWAY DEC 87

UNCLASSIFIED   F/G 12/9   NL


1·0
2·8
3·15
3·5
4·0
4·5
5·0
2·2
2·0
1·1
1·8
1·25
1·4
1·6

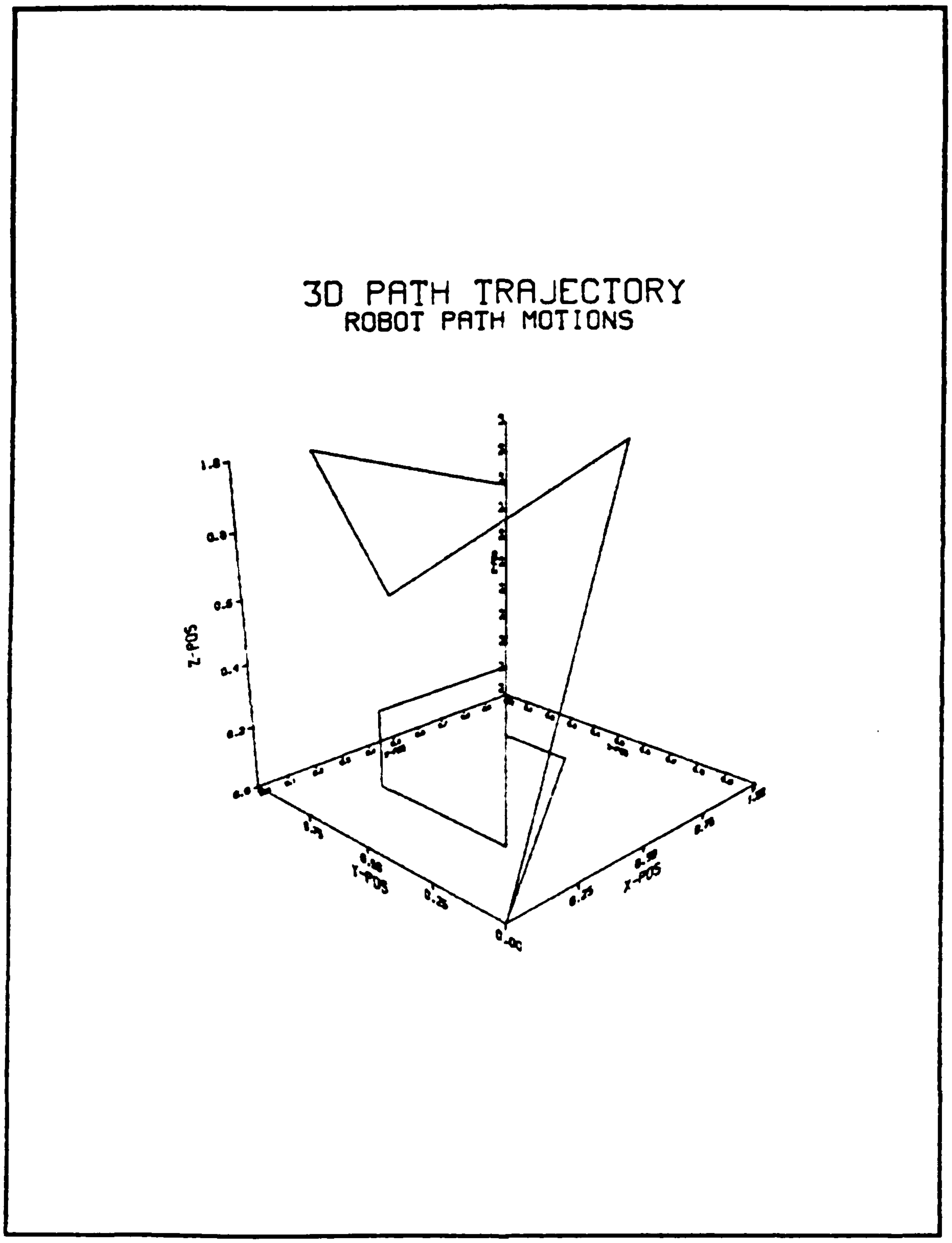


Figure 7.7(c) Articulated Robot Linear Path 2 in 3-Dimensions. [PATH3.DATA]. Pre-simulation.

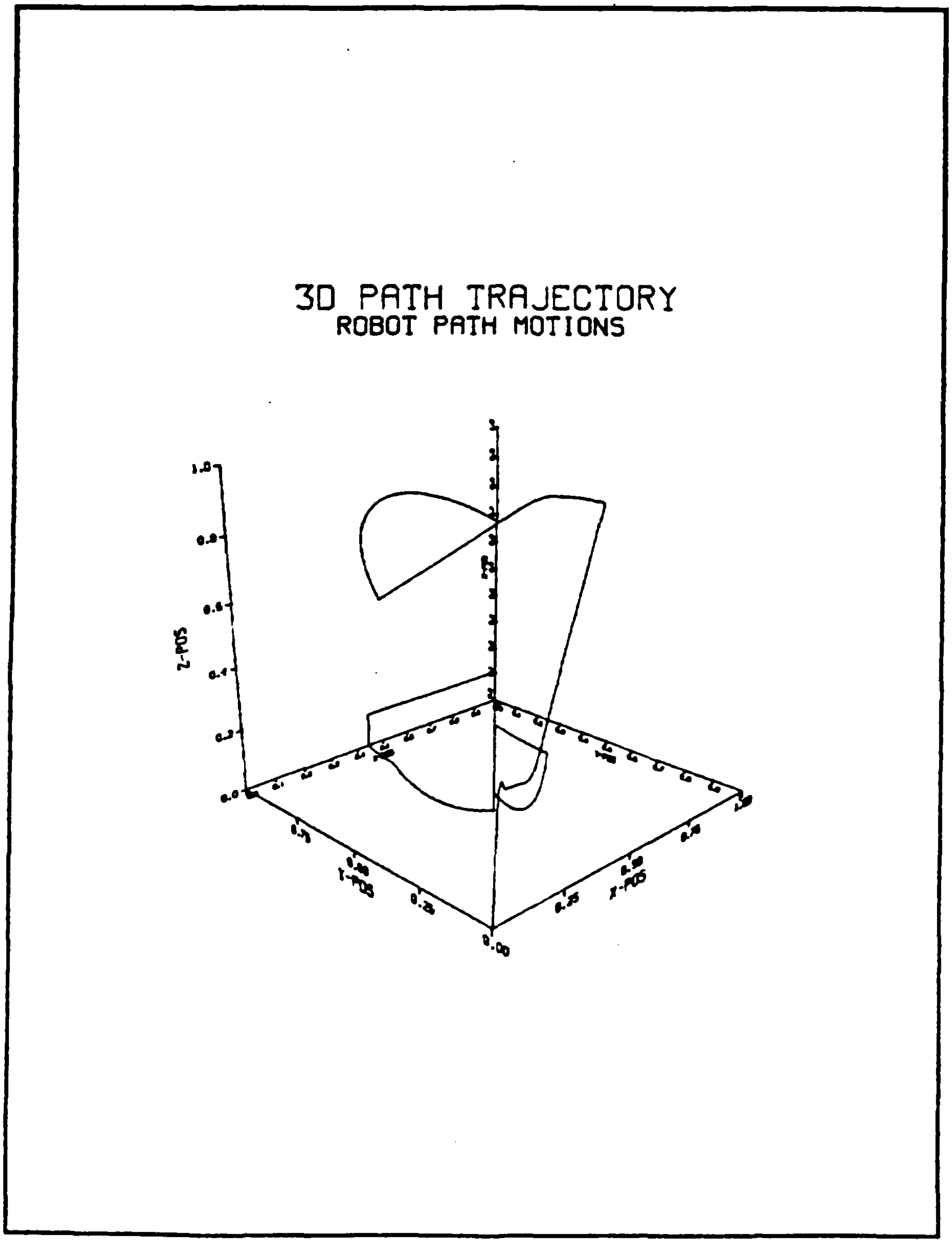


Figure 7.7(d) Articulated Robot Linear Path 2 Direct Conversion Path in 3-Dimensions. [SCPATH3.DATA]. No Simulation.

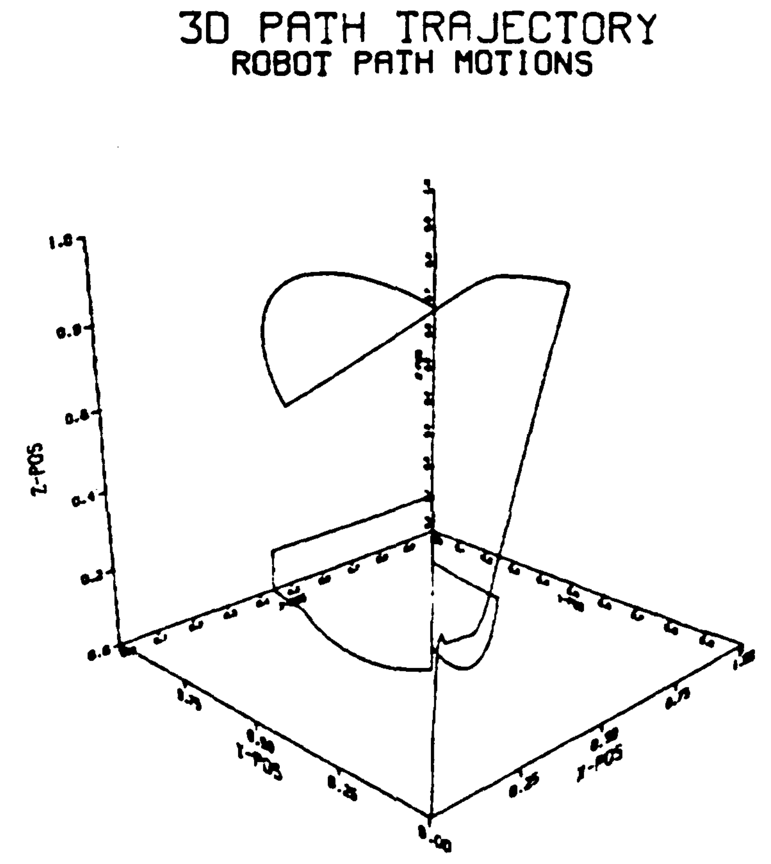


Figure 7.7(e) Articulated Robot Linear Path 2 Simulated Conversion Path in 3-Dimensions. [DCPATH3 DATA]. Post Simulation.

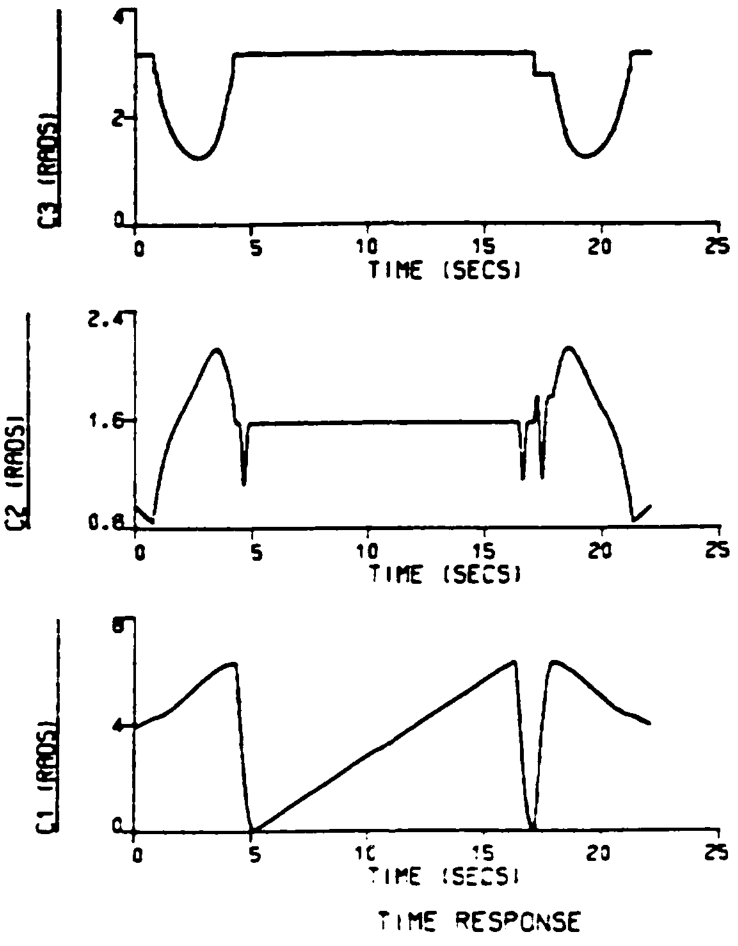


Figure 7.8(a) Articulated Robot Circular Path Time Response.

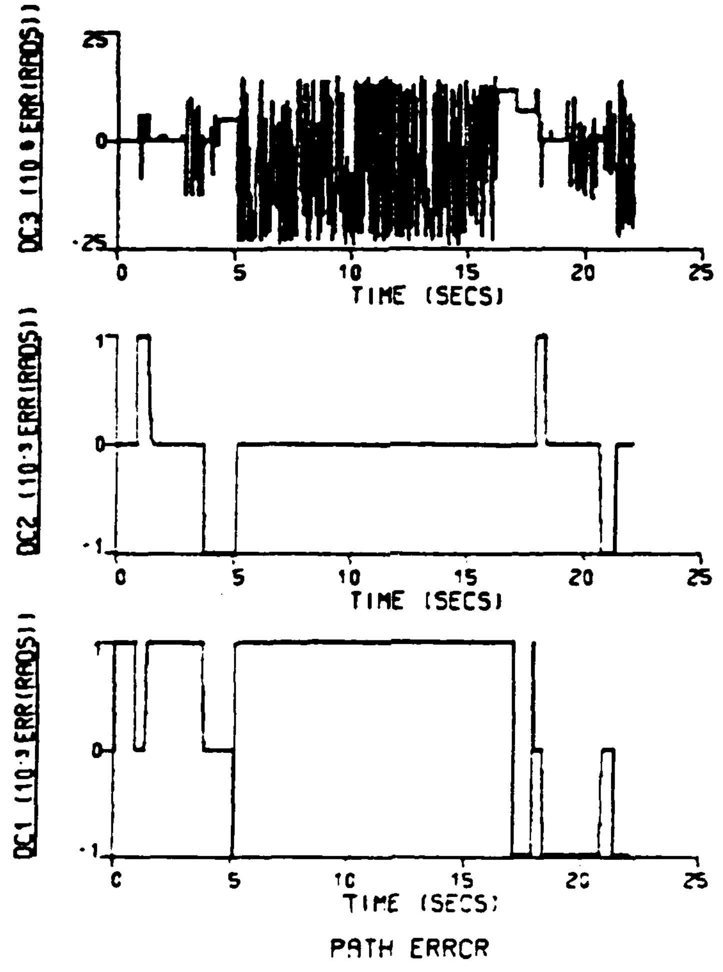


Figure 7.8(b) Articulated Robot Circular Path Error.

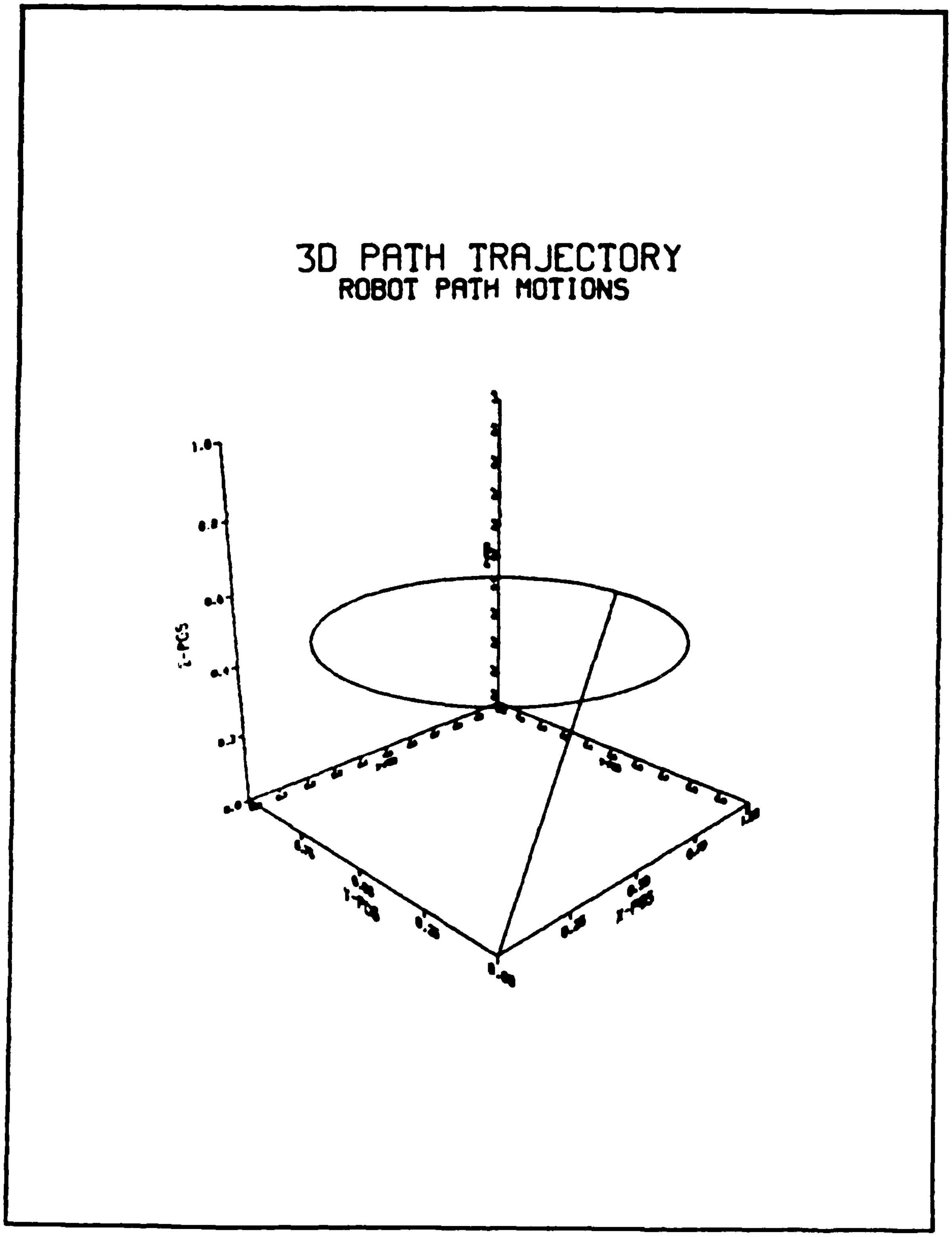


Figure 7.8(c) Articulated Robot Circular Path in 3-Dimensions [PATH4.DATA]. Pre-simulation.

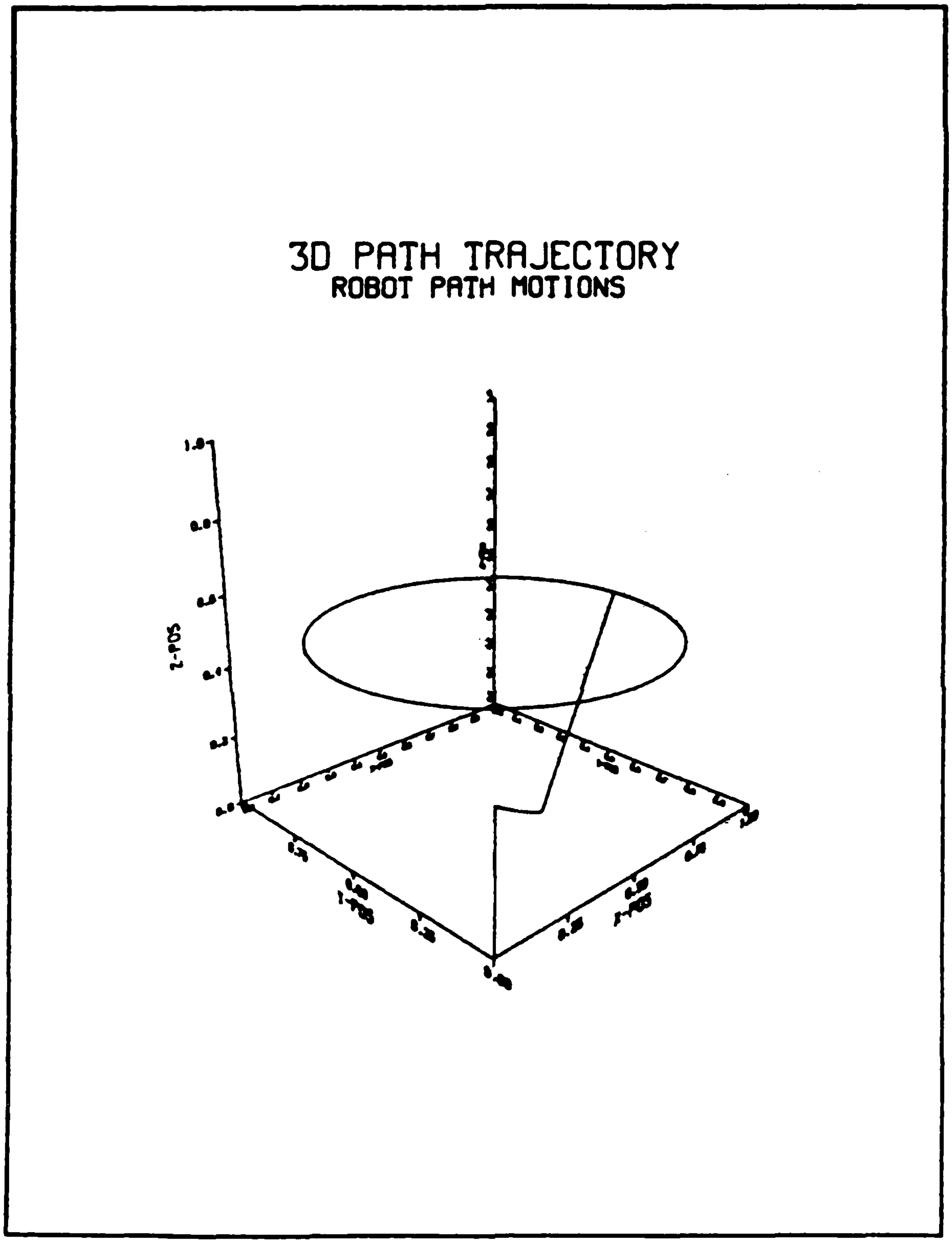


Figure 7.8(d) Articulated Robot Circular Path Direct Conversion Path in 3-Dimensions. [SCPATH4.DATA]. No Simulation.

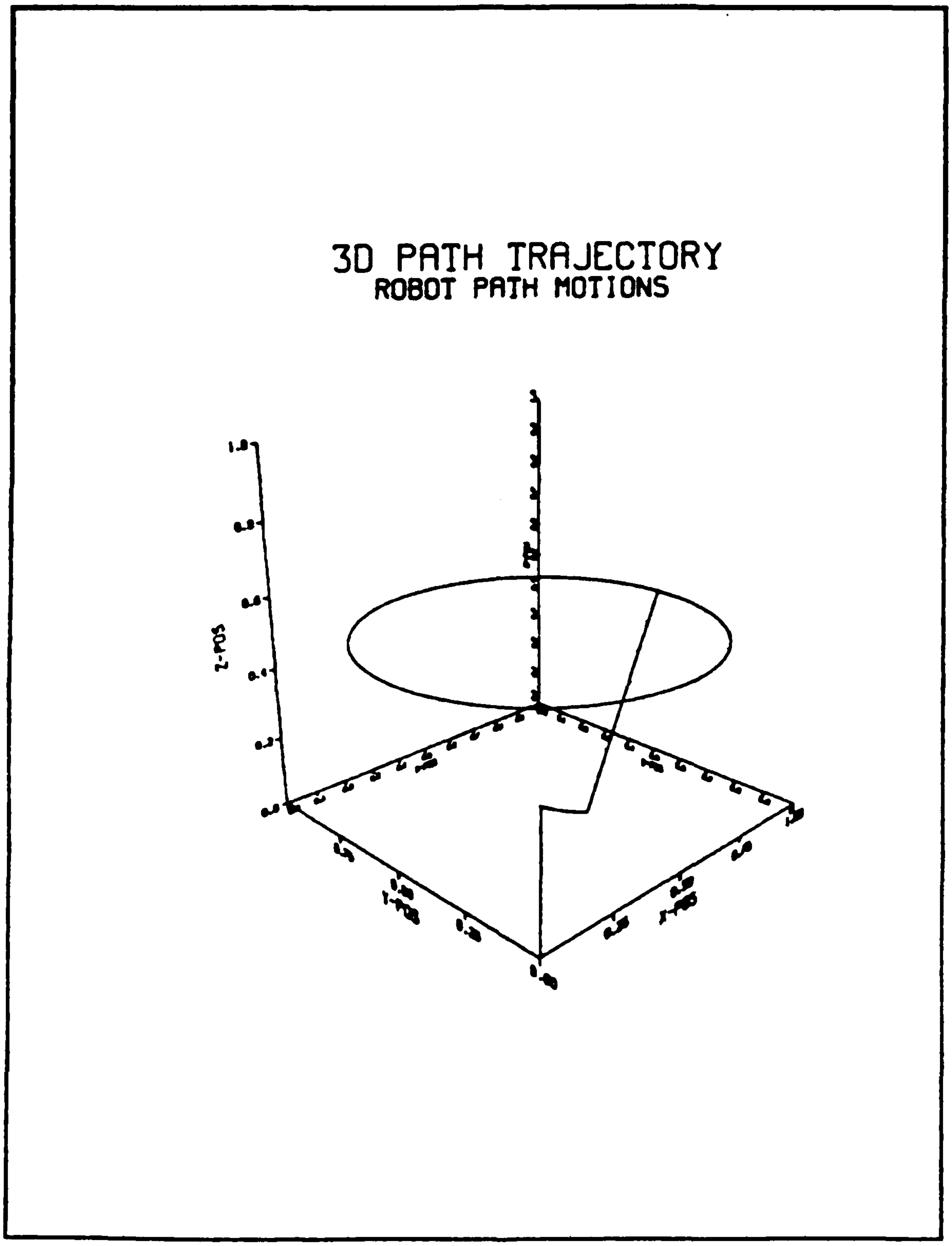


Figure 7.8(e) Articulated Robot Circular Path Simulated Conversion Path in 3-Dimensions. [DCPATH4.DATA]. Post Simulation.

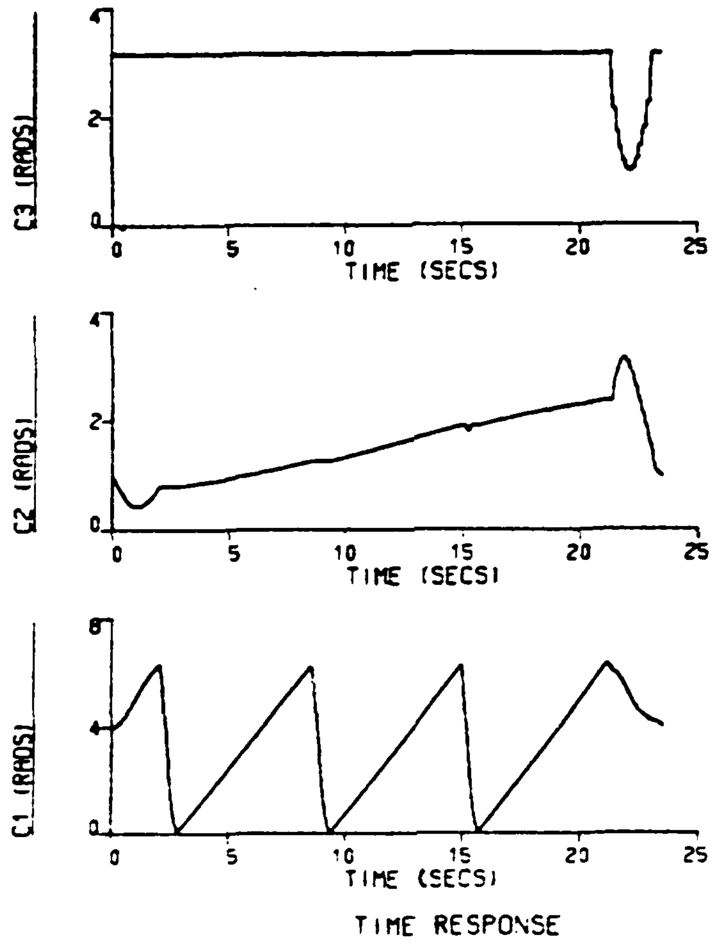


Figure 7.9(a) Articulated Robot Helix Path 1 Time Response.

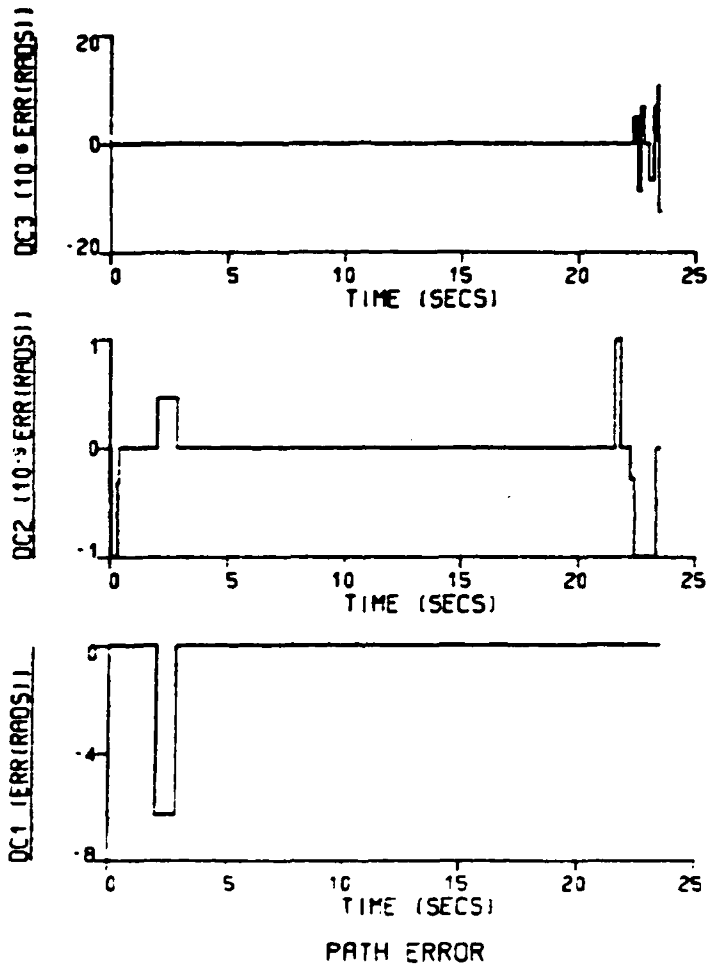


Figure 7.9(b) Articulated Robot Helix Path 1 Path Error.

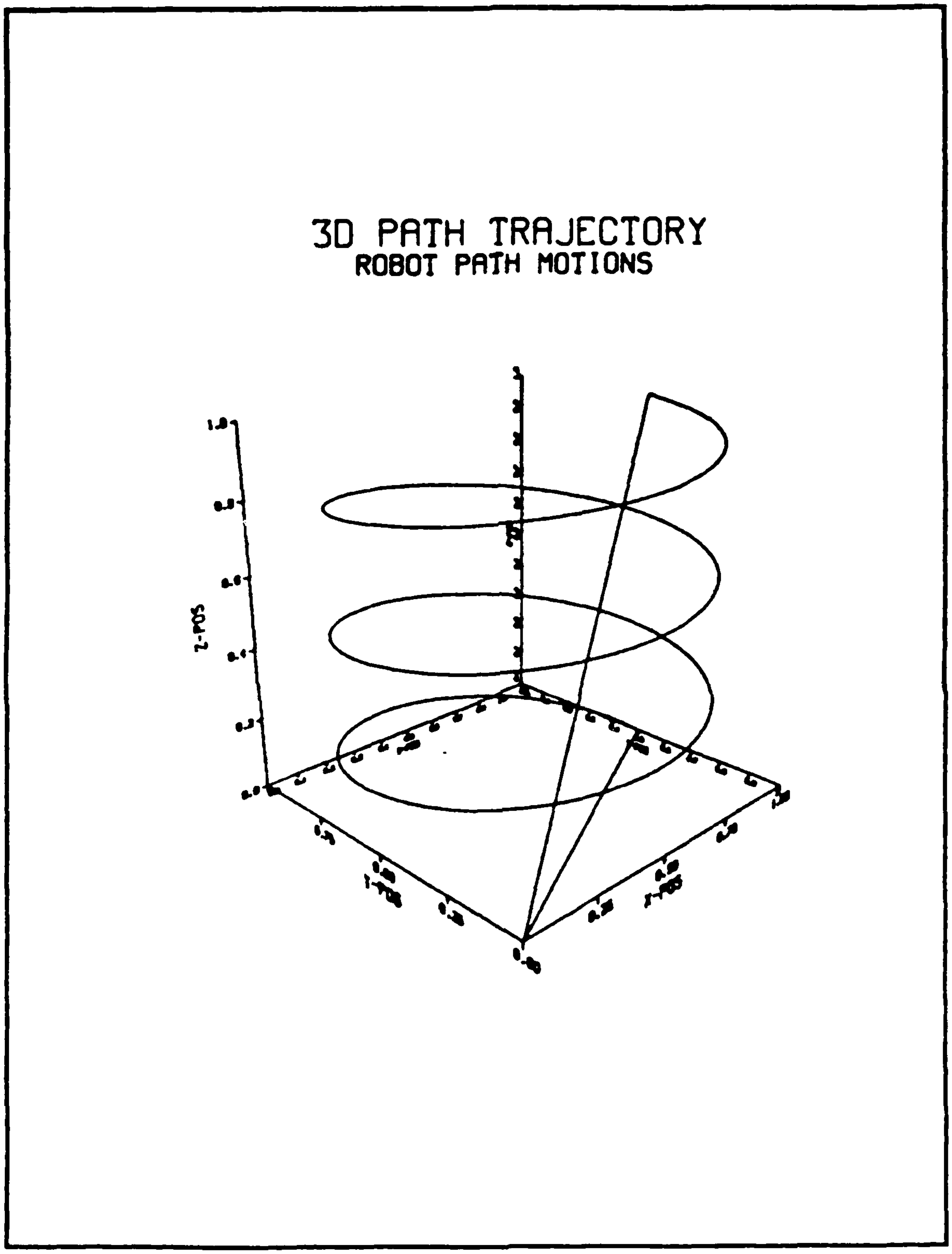


Figure 7.9(c) Articulated Robot Helix Path 1 in 3-Dimensions. [PATH5.DATA]. Pre-simulation.

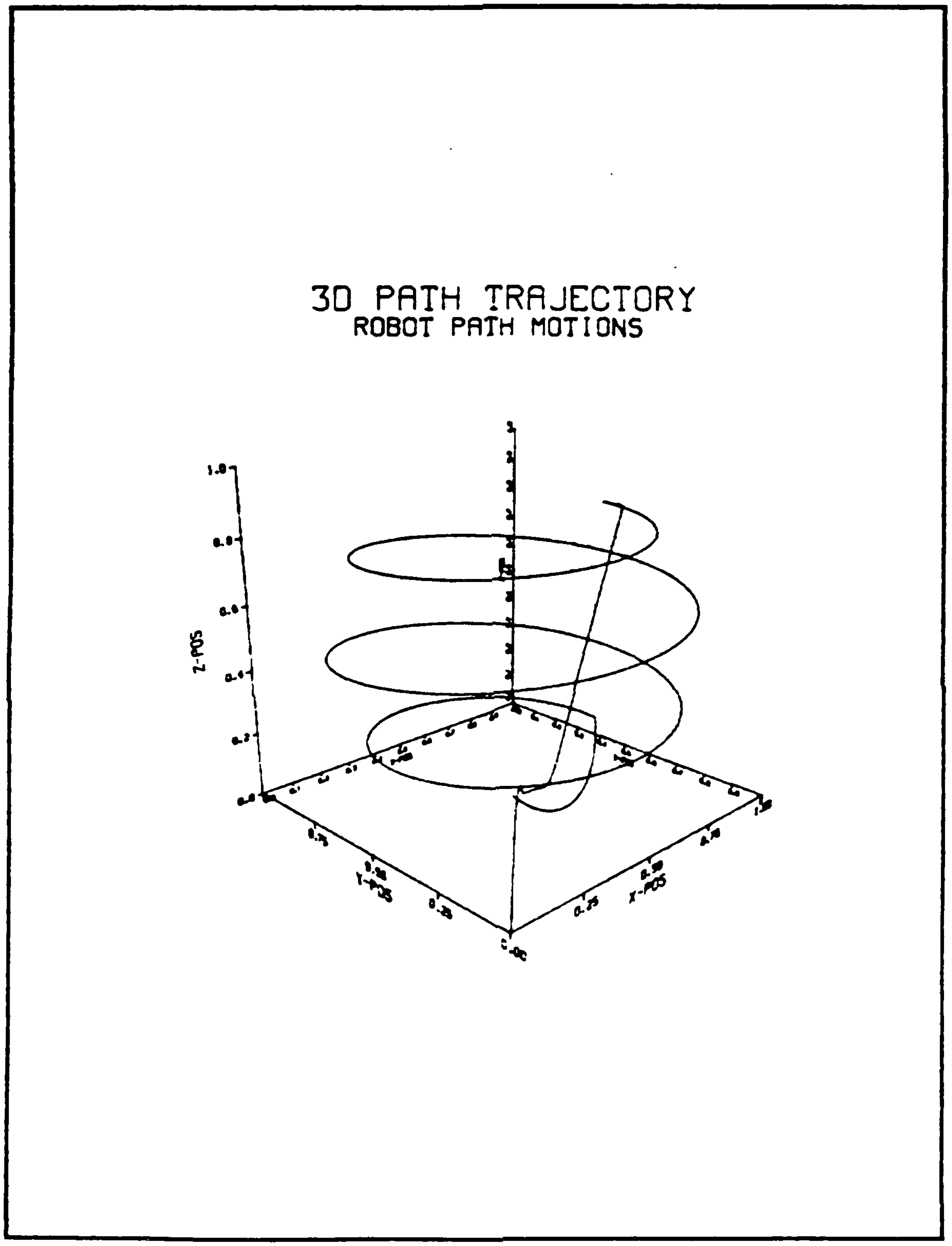


Figure 7.9(d) Articulated Robot Helix Path 1 Direct Conversion Path in 3-Dimensions. [SCPATH5.DATA]. No Simulation.

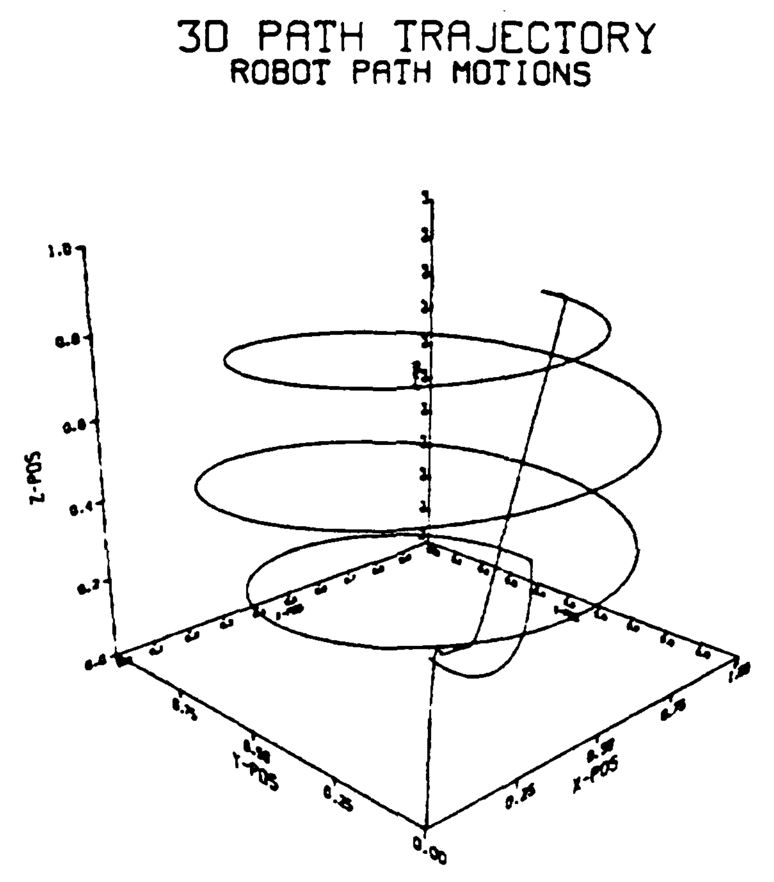


Figure 7.9(e) Articulated Robot Helix Path 1 Simulated Conversion Path in 3-Dimensions. [DCPATH5.DATA]. Post Simulation.

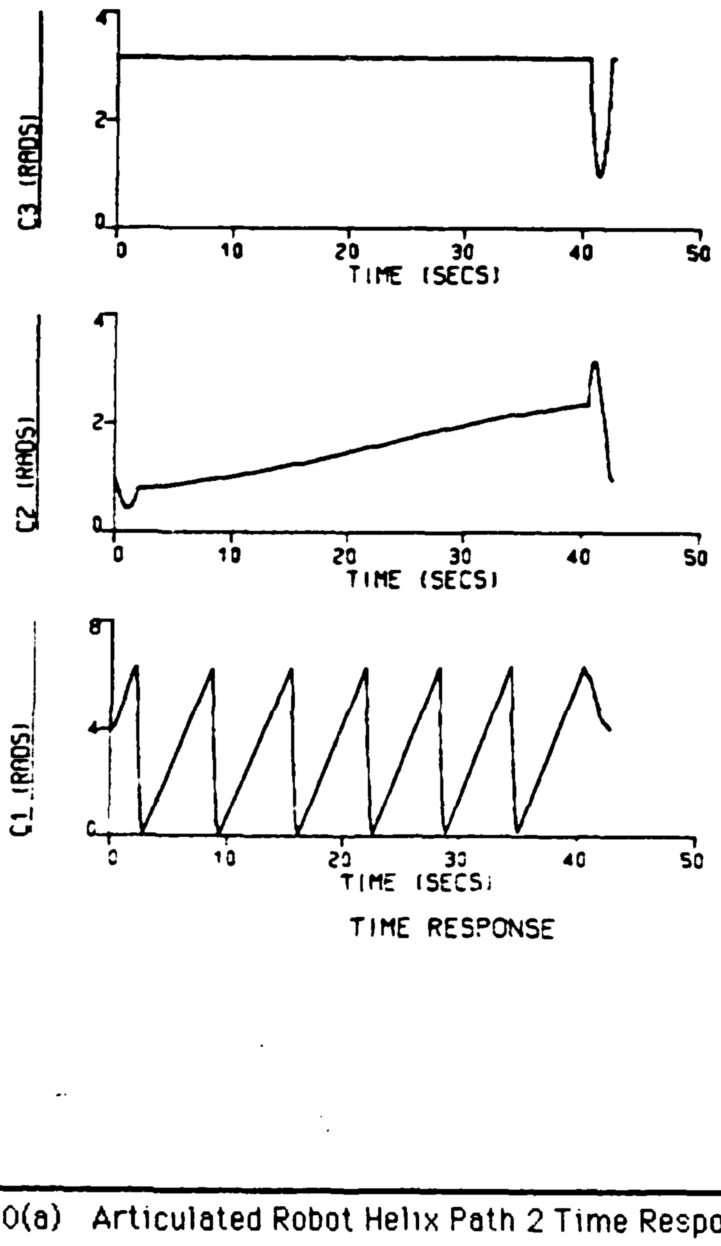


Figure 7.10(a) Articulated Robot Helix Path 2 Time Response.

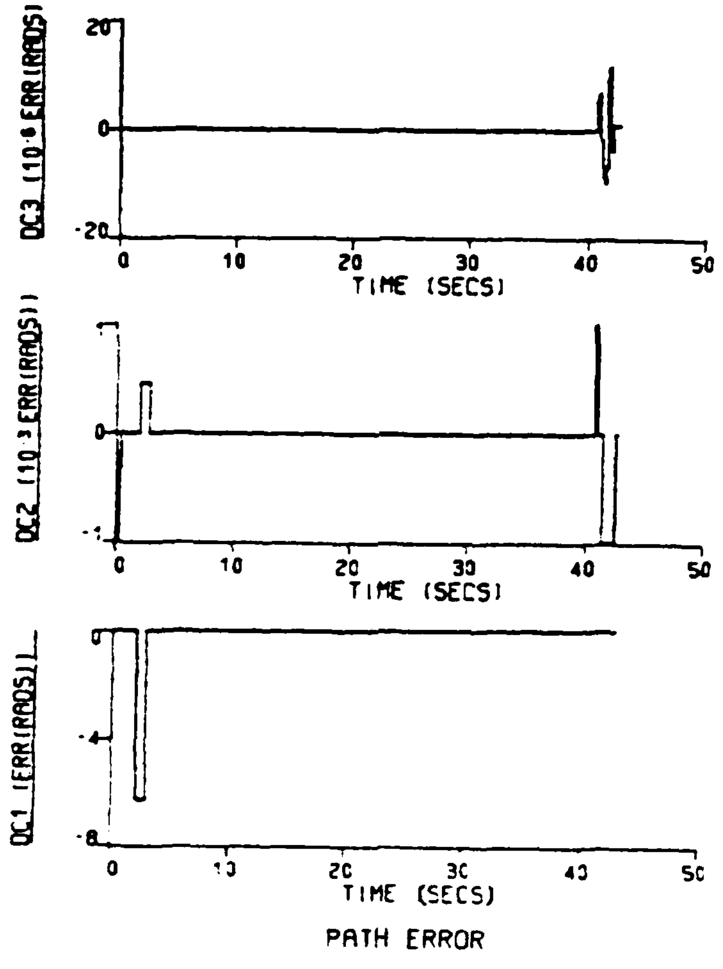


Figure 7.10(b) Articulated Robot Helix Path 2 Path Error.

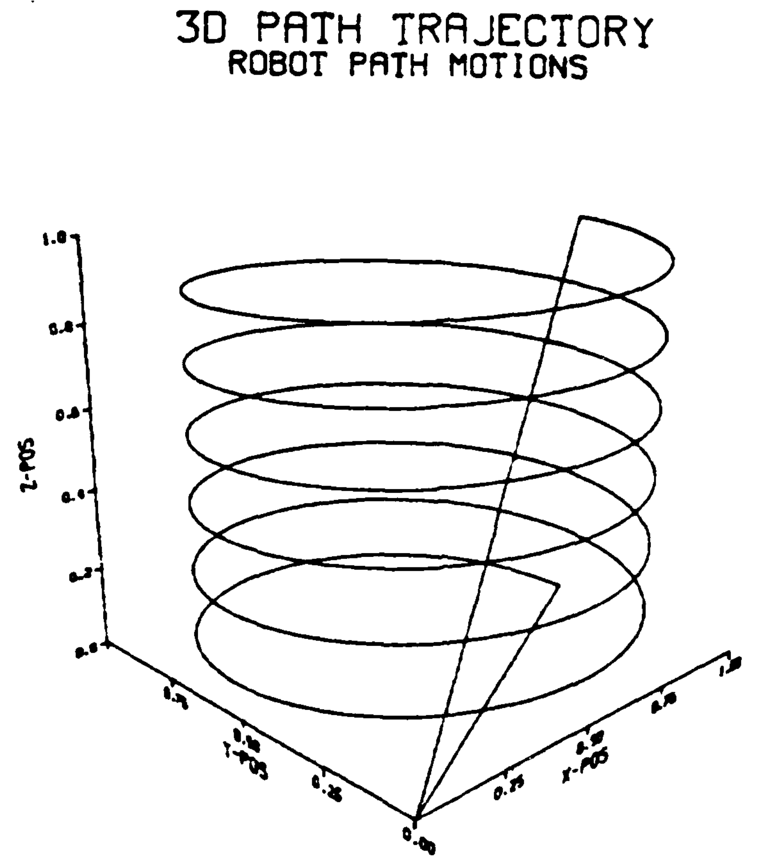


Figure 7.10(c) Articulated Robot Helix Path 2 in 3-Dimensions. [PATH5A.DATA]. Pre-simulation.

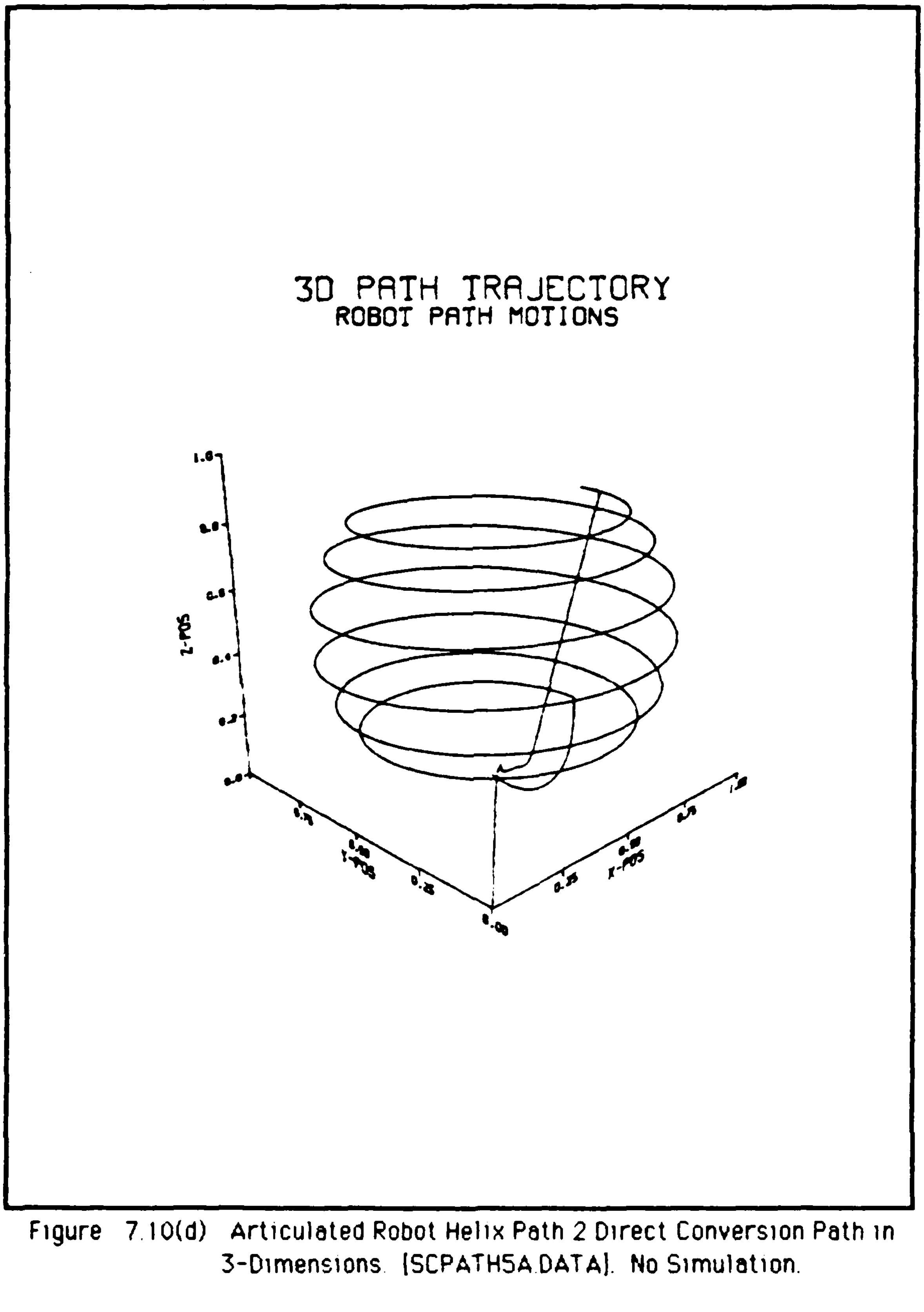


Figure 7.10(d) Articulated Robot Helix Path 2 Direct Conversion Path in 3-Dimensions. [SCPATH5A.DATA]. No Simulation.

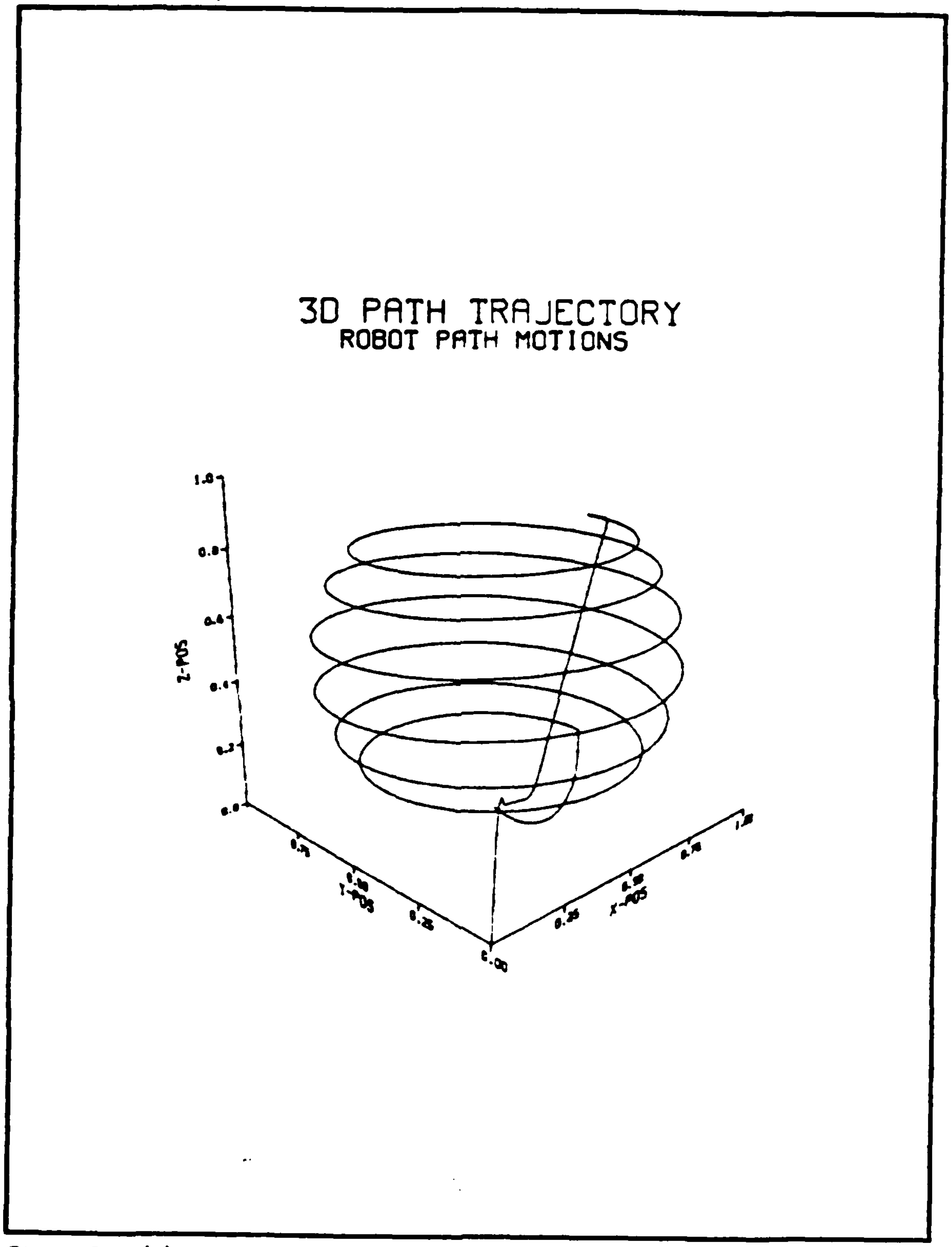


Figure 7.10(e) Articulated Robot Helix Path 2 Simulated Conversion Path in 3-Dimensions. [DCPATH5A.DATA] Post Simulation.

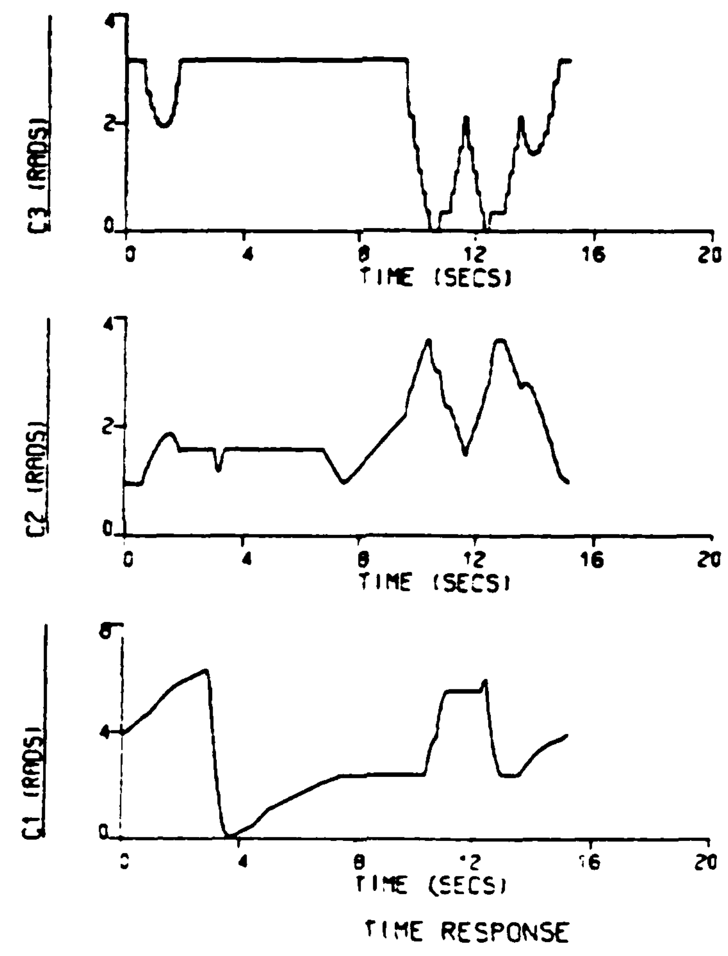


Figure 7.11(a) Articulated Robot Linear Path Time Response.

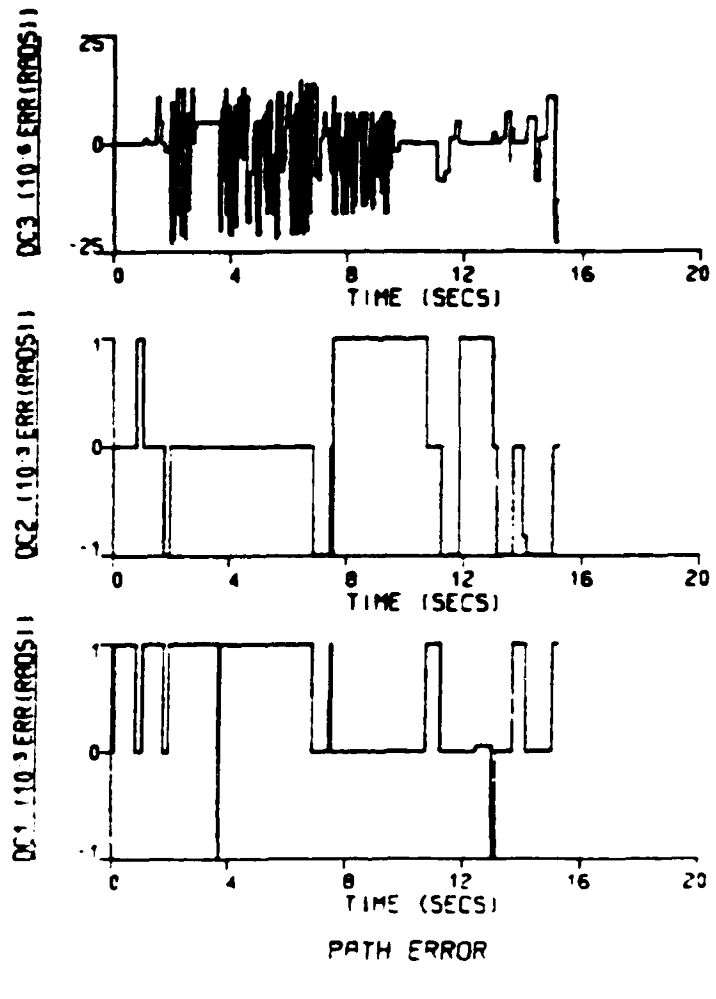


Figure 7.11(b) Articulated Robot Linear Path Error

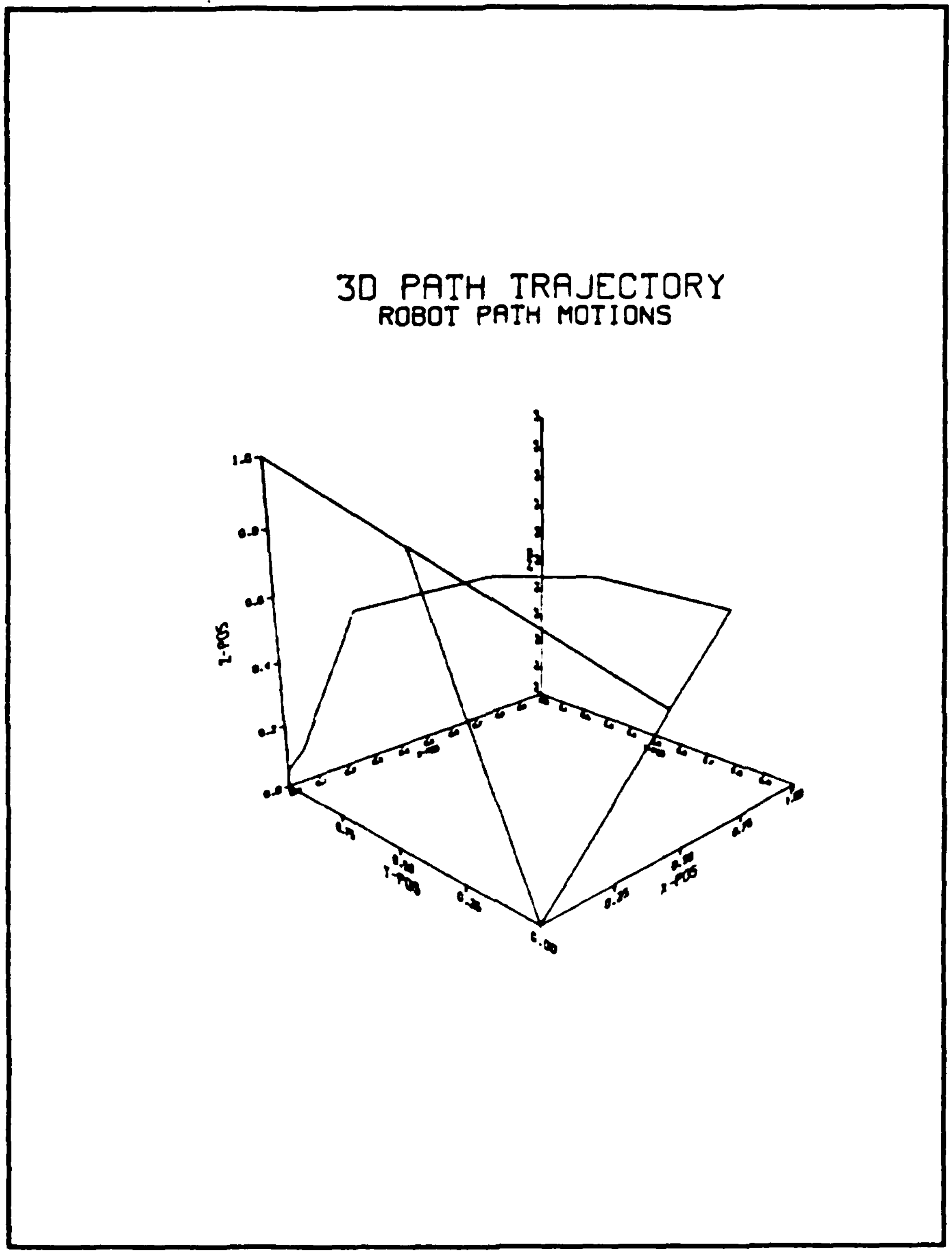


Figure 7.11(c) Articulated Robot Linear Path in 3-Dimensions [PATH6.DATA]. Pre-simulation.

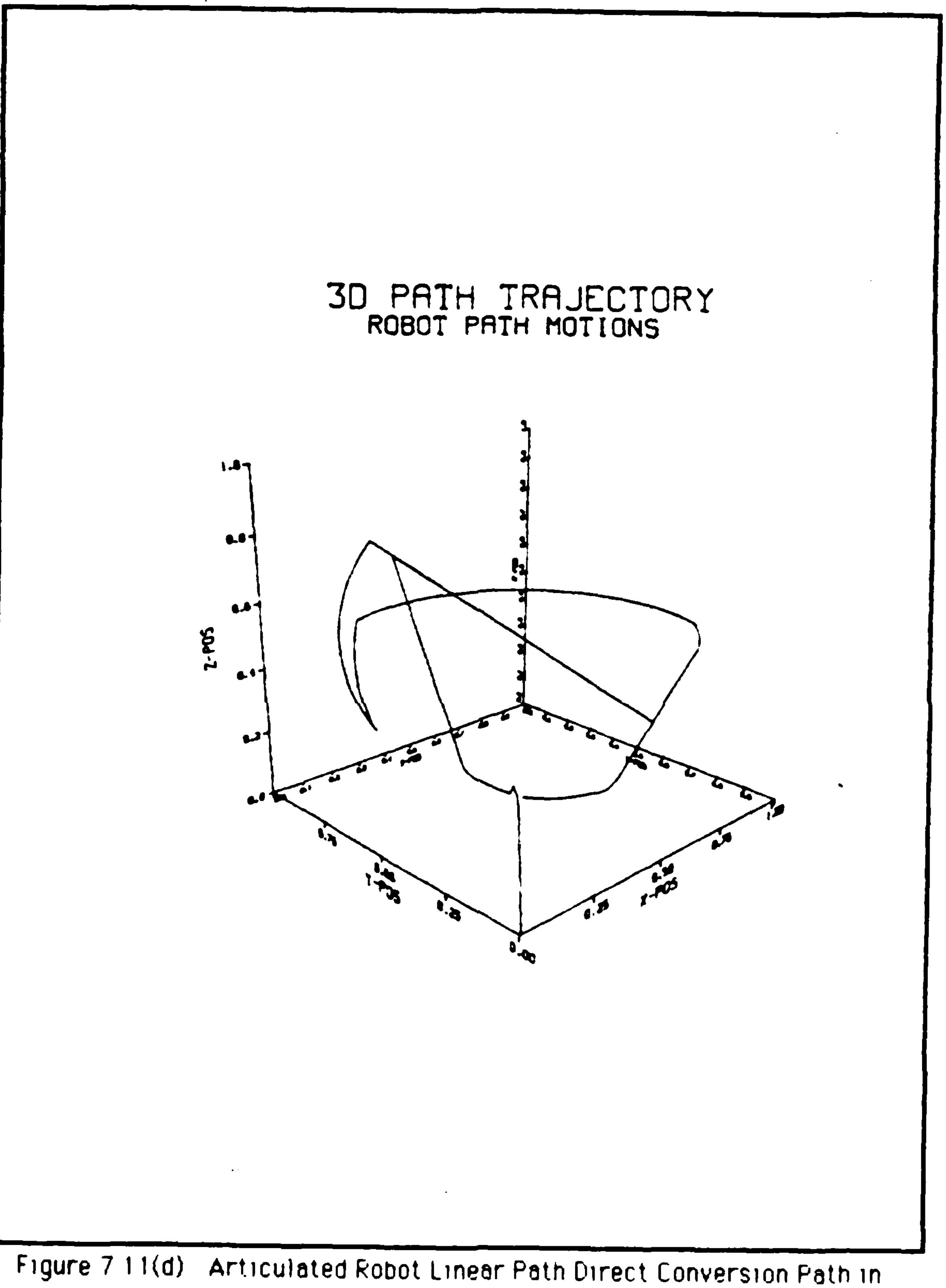


Figure 7 11(d) Articulated Robot Linear Path Direct Conversion Path in 3-Dimensions. [SCPATH6.DATA]. No Simulation.

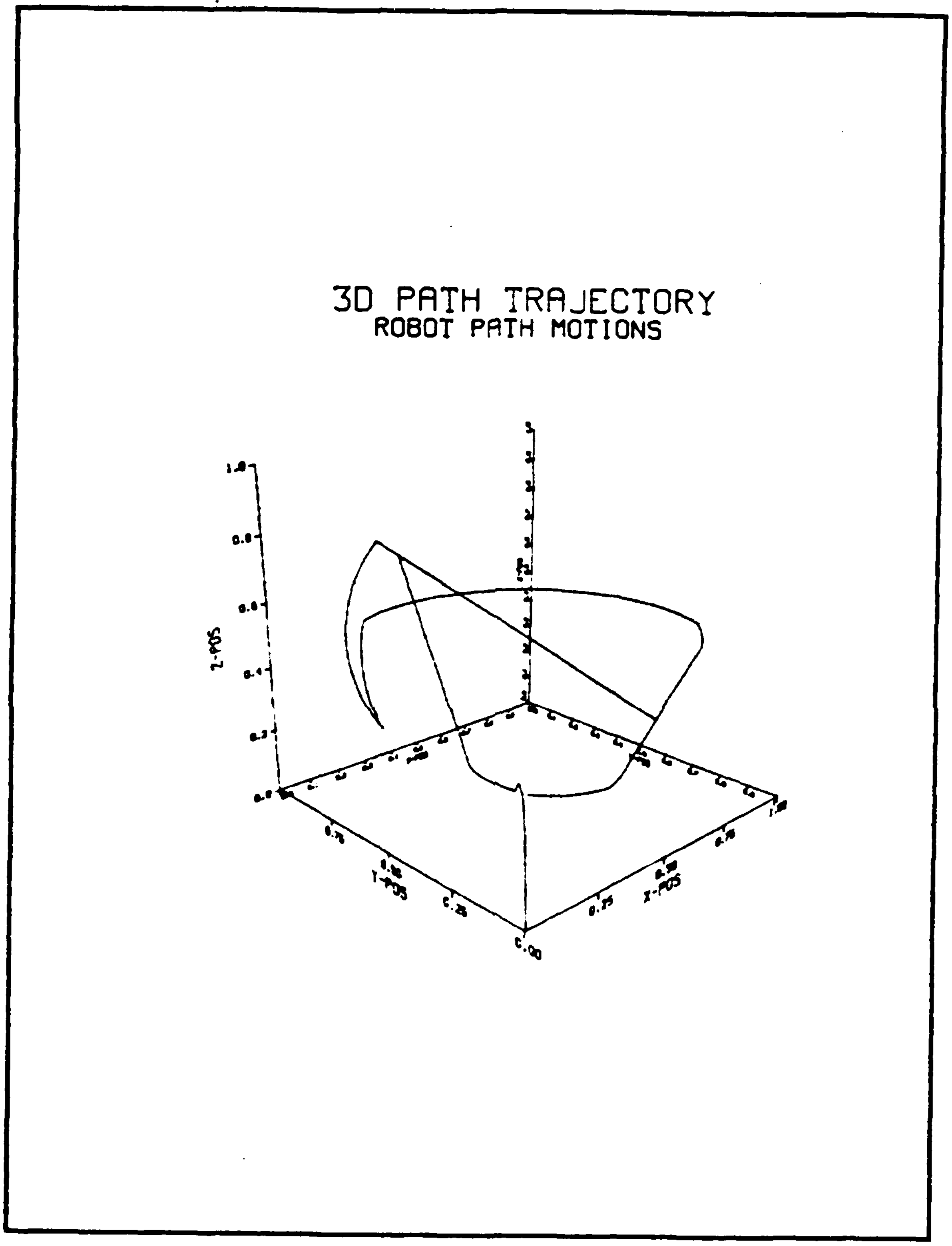


Figure 7 1 1(e) Articulated Robot Linear Path Simulated Conversion Path in 3-Dimensions. [DCPATH6.DATA]. Post Simulation.

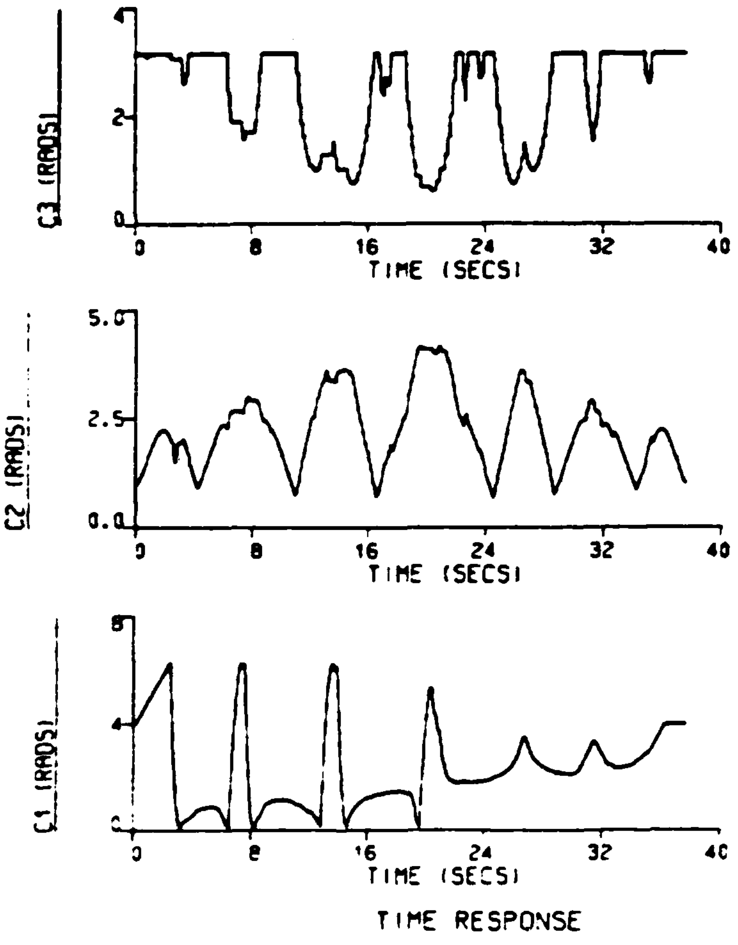


Figure 7.12(a) Articulated Robot Sinusoidal Path Time Response.

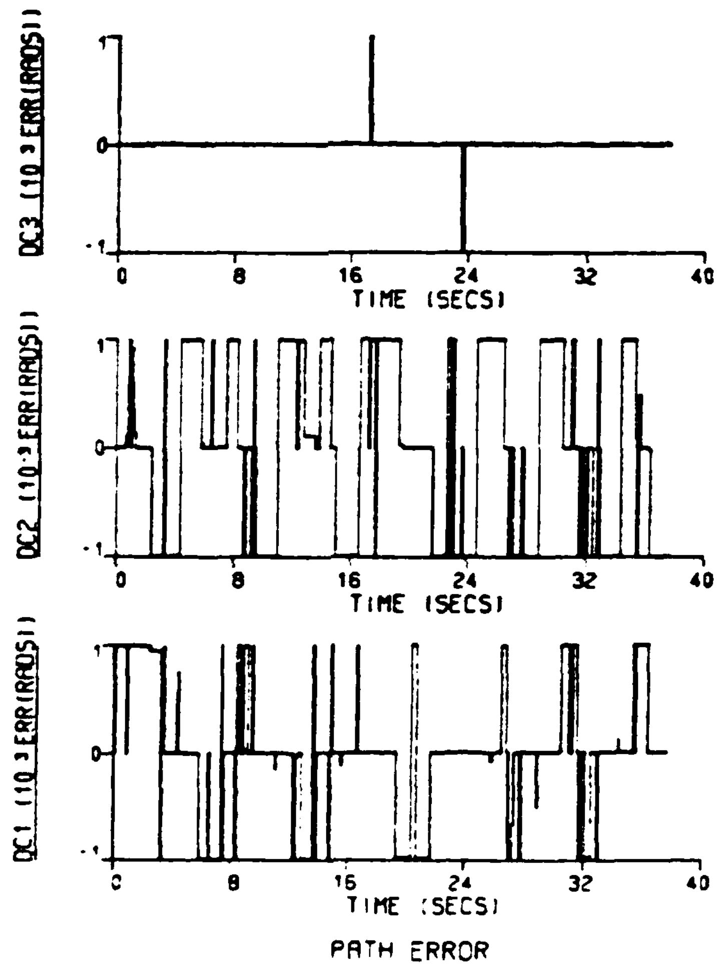


Figure 7.12(b) Articulated Robot Sinusoidal Path Error.

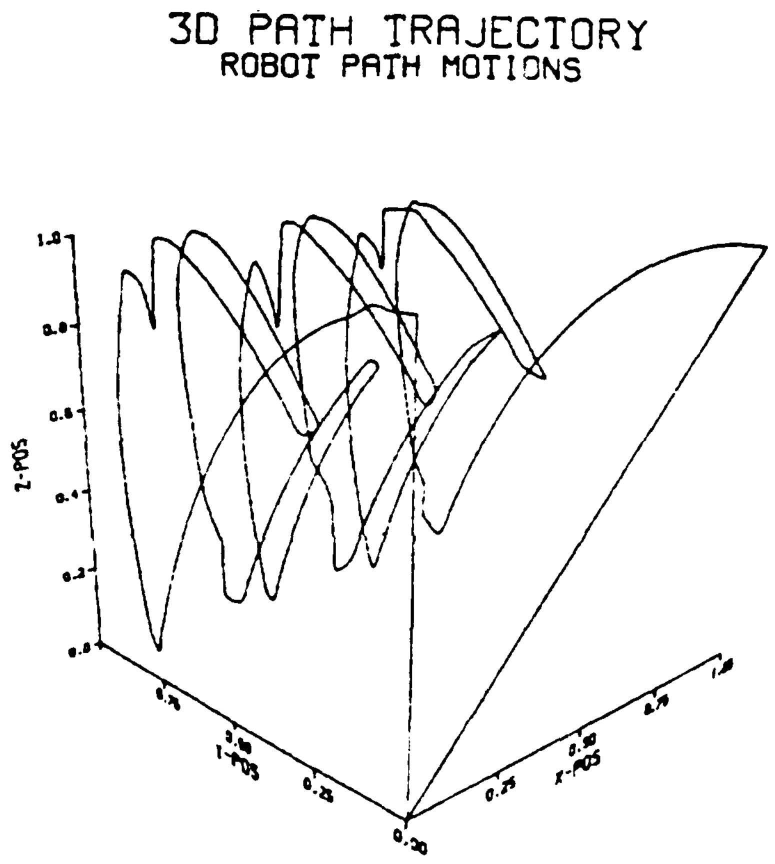


Figure 7.12(c) Articulated Robot Sinusoidal Path in 3-Dimensions. [PATH7.DATA]. Pre-simulation.

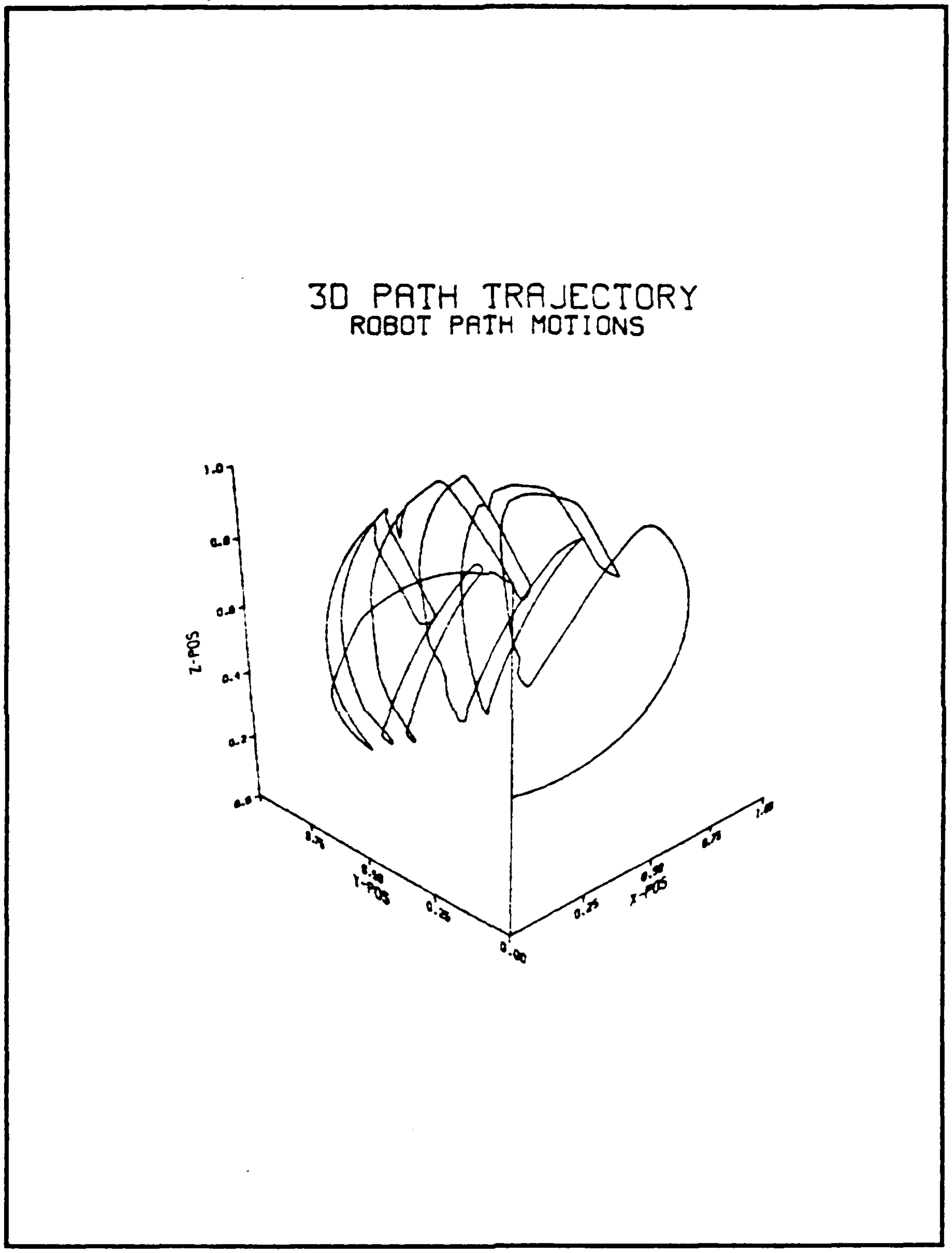


Figure 7.12(d) Articulated Robot Sinusoidal Path Direct Conversion Path in 3-Dimensions. [SCPATH7.DATA]. No Simulation.

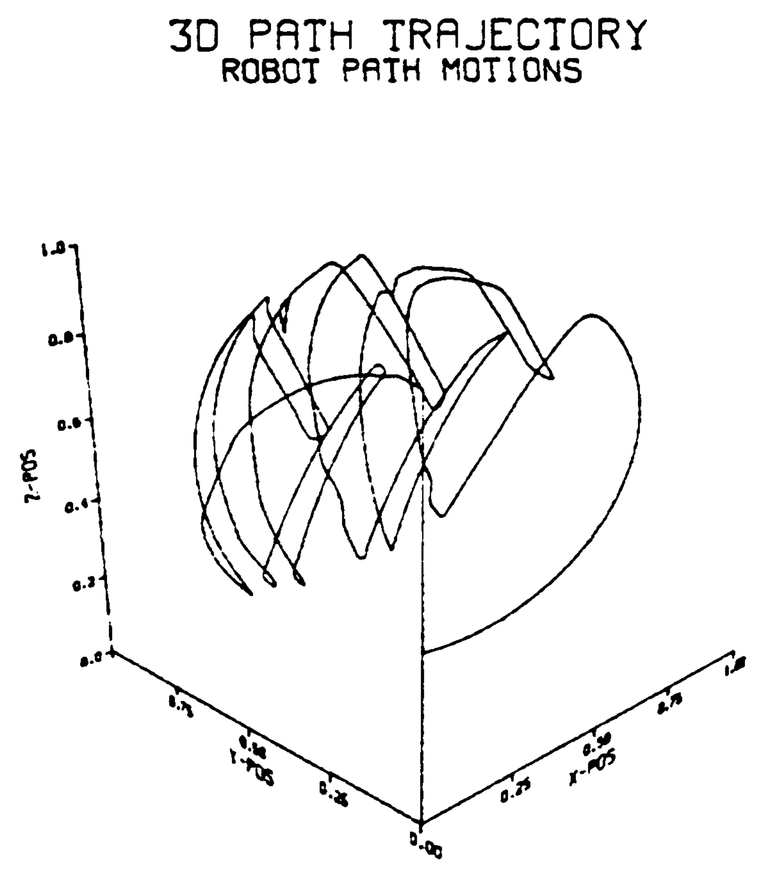


Figure 7.12(e) Articulated Robot Sinusoidal Path Simulated Conversion Path in 3-Dimensions. [DCPATH7.DATA]. Post Simulation.

VIII. CONCLUSIONS AND AREAS FOR FURTHER RESEARCH

In this thesis we have shown that it is possible to have a Robot follow a desired path accurately. Seven paths were used on both the Cartesian Robot and the Articulated Robot with good results, giving accuracies on the order of 10^{-4}.

Each of the paths were described using combinations of cartesian and parametric equations and were linked together by linear subroutines. The output of the path equations is a table of cartesian coordinates which are designed so that the maximum movement in any direction is controlled by the user (MAXOD). Also under the users control is VALMIN, which is the error between the computed position and the desired position; the accuracy (ACC), which allows higher accuracy of movement in the Position Method; the maximum velocity in each of the coordinate axes directions (MAXVX, MAXVY, and MAXVZ); and the change in time increment for the parametric routine. The user also decides on whether to plan on having STEP or RAMP Excitation to the Computer Simulation Model and the method for which the next set of points are generated. The Position Method is based on the accuracy of $E = R - C$ (Figure 3.2) being less than the value of VALMIN. When all of the d.o.f. meet that condition the the next position is inputted. The Velocity Method ignores accuracy and inputs the next position at a set time interval. The advantage of the Velocity Method is that you can get the Robot to move faster with only a slight loss in position accuracy and since one of the leading assumptions made was that the maximum error in any step is less than the distance travelled during the step then the Velocity Method is a good choice for most applications.

The Computer Model assumes that a saturated condition may be reached during the generation of the voltage that is driving ideal joint servo but the nonlinearities introduced do not significantly affect the accuracy of the desired movement and the total of the errors stay within the limit of the step size.

The programs are based on the movement space being a 1 inch cube for the Cartesian Robot, and a 1 inch diameter sphere for the Articulated Robot. With maximum errors of 10-4 inches there may not be enough accuracy for some applications. For example applying the path to a 10 meter cubed Cartesian Robot would give accuracies on the order of 1mm. For several applications this would not be sufficient accuracy. To correct this problem it might be a better approach to set up the movement space of the Cartesian Robot to be a 1000 unit cube with step sizes being 1 unit. The problem with this is that there may be excessive amount of points generated and that there may be too much accuracy for the desired job resulting in costly computation times that are not needed. As a result the size and application of the Robot need to be considered prior to designing the path generating program.

A lot of areas need to be researched further. Among those are:

1. The affects of gravity and load on the path movement.
2. Perturbations that may occur due to shifting loads or external forces applied to the robot.
3. Sensing the Robots position accurately.

4. Obstacle avoidance with not only the tip of the Robot but all of the joints and links.
5. Attachment of a wrist onto the Robot Arm and the algorithms required to move the arm and wrist into any orientation that may be desired.
6. Generating the path with higher ordered approximations. The current path generating program just uses linear techniques and there may be higher accuracies with more sophisticated techniques.
7. Using more accurate techniques for the dynamic and kinematic equations of motion. The lagrangian may not supply sufficient accuracy for higher d.o.f. than 3.
8. Optimize the program so that the program operates faster. Use of matrices may supply more speed. Current simulation requires high cpu time usage and may prove a limiting factor for higher accuracy or more complex path movements. For the Articulated Robot on a 500 point path the cpu time usage was on the order of 10's of minutes.
9. Build and demonstrate the feasibility of a Robot using only position to give the desired accuracy results, without using lead-through teaching, manual teaching, or complex programming techniques.

APPENDIX A
TRIGINOMETRY, VECTORS, AND MATRICES

This appendix contains a review of basic trigonometry and matrix algebra. Scalars are represented by lowercase letters, vectors by lowercase bold letters, and matrices by uppercase bold letters.

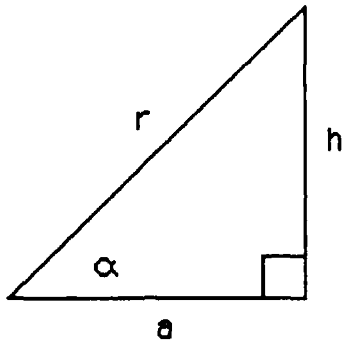
A.1 TRIGONOMETRY

Right angle relationships

$$\sin \alpha = \frac{h}{r}$$

$$\cos \alpha = \frac{a}{r}$$

$$\tan \alpha = \frac{h}{a}$$

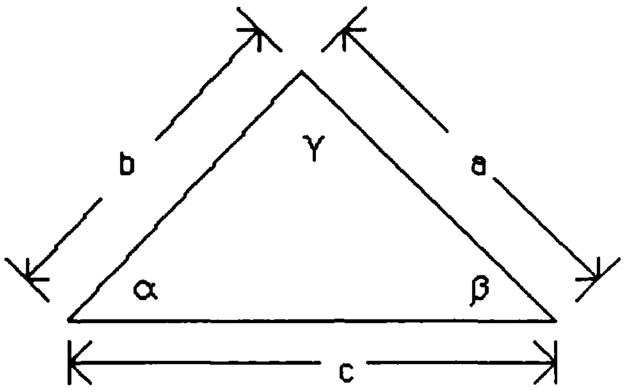


Law of Cosines

$$a^2 = b^2 + c^2 - 2bc(\cos \alpha)$$

$$b^2 = a^2 + c^2 - 2ac(\cos \beta)$$

$$c^2 = a^2 + b^2 - 2ab(\cos \gamma)$$



A.2 MATRICES

Matrix Multiplication

$\mathbf{A}$ is an n x m matrix with components $[a_{ij}]$

$i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$

$c\mathbf{A} = \mathbf{A}c = [c\, a_{ij}]$

$a(\mathbf{A} + \mathbf{B}) = a\mathbf{A} + a\mathbf{B}$

$(a + b)\mathbf{A} = a\mathbf{A} + b\mathbf{A}$

$a(b\mathbf{A}) = (ab)\mathbf{A}$

$1(\mathbf{A}) = \mathbf{A}$

$\mathbf{IA} = \mathbf{AI} = \mathbf{A}$ where I is the identity matrix and $m = n$

Given three matrices $\mathbf{A}_{mxn}$, $\mathbf{B}_{nxp}$, and $\mathbf{C}_{mxp}$

$\mathbf{A}_{mxn}(\mathbf{B}_{nxp}) = \mathbf{C}_{mxp}$ where $c_{ij} = w_k\ a_{ik}\, b_{kj}$, $k=1,2,\ldots,n$

$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$

$(\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$

$\mathbf{C}(\mathbf{A} + \mathbf{B}) = \mathbf{CA} + \mathbf{CB}$

If $\mathbf{A}$ is a square matrix and $\mathbf{A}_{ij}$ is a cofactor in $|\mathbf{A}|$

$\mathbf{A}\,\mathbf{A}^{-1} = \mathbf{A}^{-1}\,\mathbf{A} = \mathbf{I}$

$\mathbf{A}^{-1} = [\,\mathbf{A}_{ij}]^T / |\mathbf{A}| = \text{adj}\,\mathbf{A} / \mathbf{A}$

$(\text{adj}\,\mathbf{A})\,\mathbf{A} = \mathbf{A}\,(\,\text{adj}\,\mathbf{A}) = |\mathbf{A}|\,\mathbf{I}_n$

$\mathbf{A}^{-1} = 1/|\mathbf{A}|$

If $\mathbf{A}_1\mathbf{A}_2\ldots\mathbf{A}_n$ is the product of square matrices then

$(\mathbf{A}_1\mathbf{A}_2\ldots\mathbf{A}_n)^T = \mathbf{A}_n^{-1}\mathbf{A}_{n-1}^{-1}\ldots\mathbf{A}_2^{-1}\mathbf{A}_1^{-1}$

If $\mathbf{A}_1\mathbf{A}_2...\mathbf{A}_n$ is conformable then

$$(\mathbf{A}_1\mathbf{A}_2...\mathbf{A}_n)^T = (\mathbf{A}_n)^T(\mathbf{A}_{n-1})^T...(\mathbf{A}_2)^T(\mathbf{A}_1)^T$$

Translation by a vector $a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$

$$\mathrm{Trans}(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the X, Y, or Z axis

$$\mathrm{Rot}(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Rot}(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Rot}(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Transformation

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\mathbf{p}\cdot\mathbf{n} \\ o_x & o_y & o_z & -\mathbf{p}\cdot\mathbf{o} \\ a_x & a_y & a_z & -\mathbf{p}\cdot\mathbf{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Stretching Transformation

$$T = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Stretches:
X by a
Y by b
Z by c

Scaling Transformation

$$S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Paul (1981, pp. 9-62) or Fu (1987, pp. 522-55) have good sections on matrix operations as they relate to Robotics.

APPENDIX B

DENAVIT-HARTENBERG NOTATION

B.1 DENAVIT-HARTENBERG NOTATION

Denavit and Hartenberg (D-H) proposed a system designed to relate rotational and translational matrices between adjacent linkages. Attaching a coordinate system to each link the D-H representation results in a 4 x 4 matrix at each joint and 4 x 4 matrices used to transform the matrices between coordinates. In this way the end effector in "tip coordinates" can be expressed in terms of the "base coordinates" by sequentially applying the transformation matrices. Several authors (Paul, pp. 41-63; Fu, pp. 36-41; Featherstone, pp. 35-45; and Lee, pp. 68-74) go into some detail in relating transformations and the D-H for several robot models.

In Chapter Three motion was described using D-H techniques because ultimately the motion of each joint will have to be known for object avoidance. Since we have determined where we want the tip or end effector to be, it would be more convenient to use D-H matrices to reduce all of the other joint positions. Since the objective of this thesis was to follow a path without regard to the joint positions, other than the end effector, we have ignored calculating the position of the other joints, however, the D-H technique was used to generate the dynamic equations for the robot arm simulation.

B.2 ROBOT D-H MATRICES

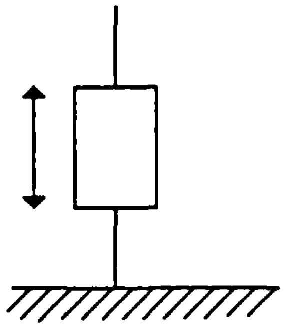
There are two types of joints, prismatic and rotary, that need to be considered. In the rotary joint the joint parameters which remain constant are d, a, α while θ changes as the link moves with respect to the previous link while for the prismatic joint the constant joint parameters are θ, a, and α, while d is variable. Figure B.1 shows an example of rotary and prismatic motion along with the associated 4 x 4 matrix. Two common robots, the Stanford Robot and the PUMA Robot, have been analysed in detail by Paul (1981, pp. 73-78), Fu (1987, pp. 37-48), Lee (1982, pp. 63-68), and Lee and Zeigler (1984, pp. 695-705).

B.3 ARTICULATED ROBOT D-H MATRICES

Figure B.2 shows the link coordinates for the 3 d.o.f. articulated robot arm and lists the link parameters and ranges of motion. The matrices are given in Figure B.3.

PRISMATIC JOINT

$$An = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & 0 \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



If θ and α are 0.0 then

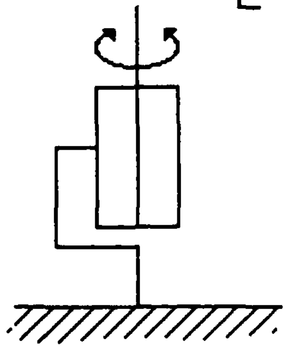
$$An = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In General:

where	α	is the twist angle (rotation around x)
	θ	is the angle rotation around z
	a	translation along rotated x
	d	translation along z

An = ROT(z,θ)TRANS(0,0,d)TRANS(a,0,0)ROT(x,α)

$$An = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



If $\theta = \theta$
$\alpha = 90$
$d = 0.0$
$a = 0.0$

then

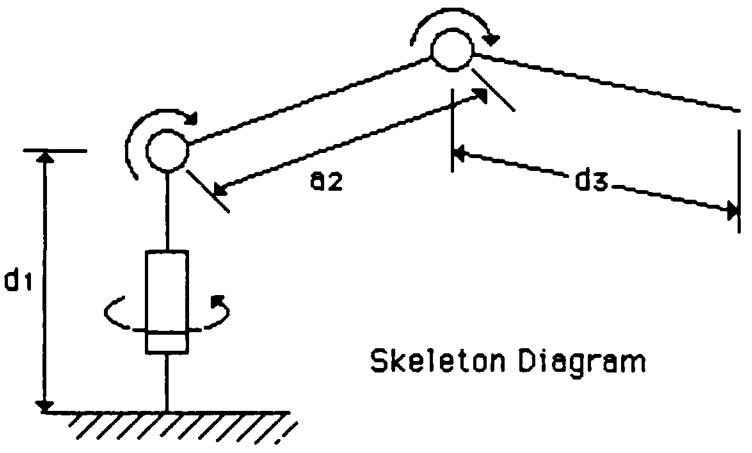
$$An = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ \sin\theta & 0 & -\cos\theta & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ROTARY JOINT

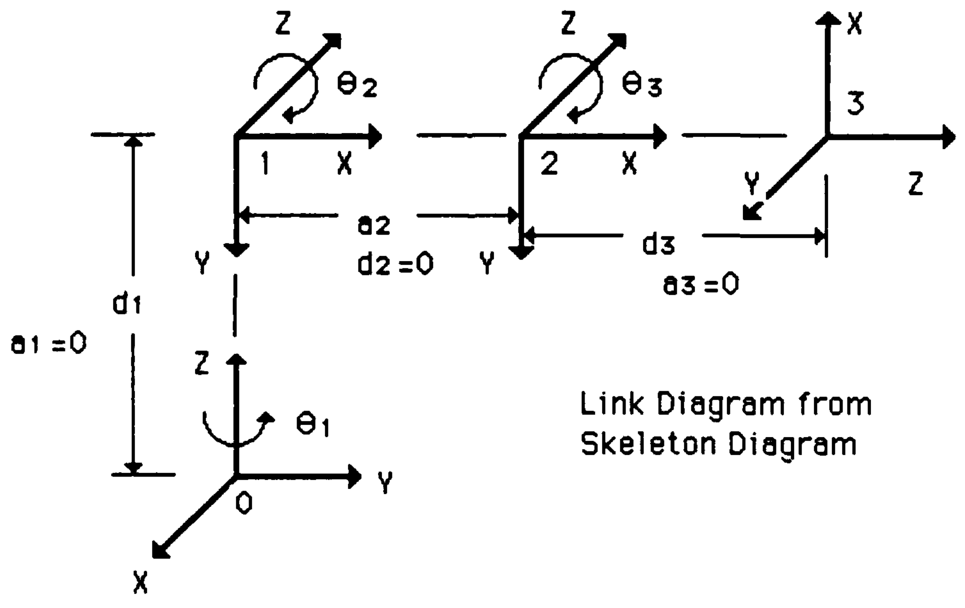
Figure B.1 Rotary and Prismatic Joints.

Link parameters for 3 d.o.f. Articulated Robot

Link	Variable	θ	α	a	d
1	θ_1	90	90	0	d_1
2	θ_2	0	0	a_2	0
3	θ_3	-90	90	0	d_3



Skeleton Diagram



Link Diagram from Skeleton Diagram

Figure B.2 Link Coordinates and Parameters for the Articulated Robot Model.

$$
{}^{0}A_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}A_2 = \begin{bmatrix} \cos\theta 2 & -\sin\theta 2 & 0 & a2\cos\theta 2 \\ \sin\theta 2 & \cos\theta 2 & 0 & a2\sin\theta 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}A_3 = \begin{bmatrix} \cos\theta 3 & 0 & \sin\theta 3 & 0 \\ \sin\theta 3 & 0 & -\cos\theta 3 & 0 \\ 0 & 1 & 0 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}T_3 = {}^{0}A_1 \; {}^{1}A_2 \; {}^{2}A_3
$$

$T = {}^{0}A_6$ specifies the position and orientation of the of the Robot with respect to the ground. The T matrix is called the Arm Matrix.

$$
T = {}^{0}T_3 \; {}^{3}T_6
$$

Figure B.3 Articulated Robot D-H Matrices

APPENDIX C
CARTESIAN ROBOT DSL PROGRAMS

This appendix contains the Cartesian Robot DSL programs used to model the Cartesian Robot Arm. One program is presented with four variations that were used.

ROBSIM2T.FORTRAN / RSIM2T6.FORTRAN

These programs are identical except in the way that the graphs are plotted. ROBSIM2T plots each graph on a seperate page, while RSIM2T6 plots up to six graphs on a single page. Modifications to the Robot Simulation [Kalagarios] program allow for inputting multiple data runs, including zero steps, and can be adapted for either STEP or RAMP Excitation as desired. Inputs are controlled on a position accuracy basis.

ROBSIM2V.FORTRAN / RSIM2V6.FORTRAN

These programs are identical to the previously mentioned programs except in the way that the data points are inputted. Inputs are controlled on a time basis and is reflected in two line changes of the ROBSIM2T program.

ROBSIM2T.FORTRAN LISTING

```
TOF:
C     STEVEN G. GOODWAY
C     ROBSIM2T FORTRAN
C     NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C     CARTESIAN ROBOT SIMULATION {ROBSIM2T.FORTRAN}
C     THIS PROGRAM TAKES A THREE TUPLE OF POINTS (X,Y,Z) FROM A
C     TABLE OF THREE TUPLES AND APPLIES THEM TO SIMULATION OF A
C     CARTESIAN ROBOT. THE RESULTING MOVEMENT FROM THE SIMULATION
C     IS THE EXPECTED MOVEMENT OF THE TIP OR ENDPOINT OF THE ROBOT.
C     THE NEXT THREE TUPLE OF POINTS ARE INPUTTED AFTER THE VALUE
C     OF THE ERROR APPLIED TO ALL OF THE MOTORS HAS DECREASED TO
C     BELOW A VALUE CALLED VALMIN. THERE IS ONE BASIC ASSUMPTION
C     THAT WAS MADE WHEN CONSTRUCTING THE ORIGINAL THREE TUPLE TABLE
C     AND WAS CHOSEN TO INSURE THE ACCURACY WHEN APPLYING THE POINTS
C     TO THE SIMULATION.
C
C     ASSUMPTIONS:
C
 C    1.        THE MAXIMUM ERROR DURING ANY MOVE IS LESS THAN OR EQUAL
 C              TO THE DISTANCE TRAVELED IN THAT DIMENSION DURING THE
 C              STEP.
C
C               FILEDEFS REQUIRED FOR PROPER OPERATION OF THE PROGRAM ARE
 C              FILEDEF 02  DISK  PATHX   DATA
 C              FILEDEF 03  DISK  SCOMPX  DATA
C
 C   GRAPHS THAT RESULT ARE SINGLE PAGE GRAPHS (9 TOTAL)
 C    TIME RESPONSE (3)
 C    PLANE PROJECTION (3)
 C    PATH ERROR (3)
C
C     ************************************
C     * SIMULATION PROGRAM FOR CARTESIAN ROBOT *
C     ************************************
C
C     MAIN PROGRAM AREA
C     USER PARAMETER ADJUSTMENTS
C
 PARAM K=1.0,K1=0.6,K2=10000.0,KM1=59.29,KM2=90.25,KM3=77.44
 PARAM VSAT=150.0,M1=0.082,M2=0.041,M3=0.041,MM=0.186
 PARAM J1=0.033,J2=0.033,J3=0.033,R1=0.91,R2=0.91,R3=0.91
 PARAM KT1=14.4,KT2=14.4,KT3=14.4,L1=0.0001,L2=0.0001,L3=0.0001
 PARAM BM1=0.04297,BM2=0.04297,BM3=0.04297,RO=0.5,LOAD=0.0
 PARAM KV1=0.1012,KV2=0.1012,KV3=0.1012,T=0.00025,VALMIN=0.001
 PARAM TV1=0.0075,TV2=0.0075,TV3=0.0075
 INTEGER SW1,SW2,SW3,N1,N2,N3,J
 *
 *    K1:              CURVE SCALING CONSTANT
*    K2:                AMPLIFIER GAIN
*    KM:                IDEAL(MODEL) MOTOR CONSTANT
*    VSAT:              SATURATION LIMITS OF AMPLIFIER
*    K:                 VELOCITY LOOP FEEDBACK GAIN (MODEL)
*    XPOS,YPOS,ZPOS:              COMMANDED ENDPOINT POSITION
*    T:                 SAMPLING INTERVAL
*    VALMIN: VALUE OF E USED TO CAUSE NEXT STEP TO BE INPUTTED.
*    H:                 NUMBER OF POINTS ALREADY INPUTTED
*    J:                 NUMBER OF POINTS IN PATH (+1)
*
*
INITIAL
     J=5
     H=0
     SW1=0
     SW2=0
     SW3=0
     N1=0
     N2=0
     N3=0
     RX=0.0
     RY=0.0
     RZ=0.0
     C1=0.0
     C2=0.0
     C3=0.0
     X1DOT=0.
     X2DOT=0.
     X3DOT=0.
     TL1=0.
     TL2=0.
     TL3=0.
     CF=1./RO
     CALL OPEND (J)
     CALL NEXPT   (RX,RY,RZ,C1,C2,C3)
DERIVATIVE
NOSORT
     A1=SQRT(2.*KM1*VSAT)
     A2=SQRT(2.*KM2*VSAT)
     A3=SQRT(2.*KM3*VSAT)
     REF1=RX*CF
     REF2=RY*CF
     REF3=RZ*CF
SORT
     TM1=M1+M2+M3+2*MM+LOAD
     TM2=M2+LOAD
     TM3=M2+M3+MM+LOAD
     JM1=TM1*RO
     JM2=TM2*RO
     JM3=TM3*RO
```

```
       JTOT1=J1+JM1
       JTOT2=J2+JM2
       JTOT3=J3+JM3
*      USE FOR STEP EXCITATION
*      RTH1=REF1*STEP(0.0)
*      RTH2=REF2*STEP(0.0)
*      RTH3=REF3*STEP(0.0)
*      USE FOR RAMP EXCITATION
       RTH1=(REF1/TV1)*(TIME-RAMP(TV1))
       RTH2=(REF2/TV2)*(TIME-RAMP(TV2))
       RTH3=(REF3/TV3)*(TIME-RAMP(TV3))
       E1=RTH1-C1
       E2=RTH2-C2
       E3=RTH3-C3
******************************************************** NOSORT
       IF (E1.LT.0.0) X1DOT=-A1*K1*SQRT(ABS(E1))
       IF (E1.GE.0.0) X1DOT= A1*K1*SQRT(E1)
       IF (E2.LT.0.0) X2DOT=-A2*K1*SQRT(ABS(E2))
       IF (E2.GE.0.0) X2DOT= A2*K1*SQRT(E2)
       IF (E3.LT.0.0) X3DOT=-A3*K1*SQRT(ABS(E3))
       IF (E3.GE.0.0) X3DOT= A3*K1*SQRT(E3)
*      USE FOR RAMP OR SINE FUNCTIONS/ POSITION METHOD
*      IF ((ABS(CX-RX).LT.VALMIN).AND.(ABS(CY-RY).LT.VALMIN) ...
*        .AND.(ABS(CZ-RZ).LT.VALMIN)) THEN
       IF ((RTH1.GE.REF1).AND.(RTH2.GE.REF2).AND.(RTH3.GE.REF3))THEN
*               USE FOR STEP FUNCTIONS/ POSITION METHOD
       IF ((ABS(E1).LT.VALMIN).AND.(ABS(E2).LT.VALMIN).AND. ...
              (ABS(E3).LT.VALMIN)) THEN
       CALL ENDRUN
*                       USE FOR VELOCITY METHOD  ***********ROBSIM2V*****
*                       IF (TIME.GE.((H+1)*TSTEP)) THEN
*      CALL ENDRUN
       ENDIF
       ENDIF
SORT
********************************************************
KC1DOT=K*C1DOT
      X1DOTE=X1DOT-KC1DOT
       V1=LIMIT(-VSAT,VSAT,K2*X1DOTE)
       C1DDT=V1*KM1
       C1DOT=INTGRL(0.0,C1DDT)
       C1=INTGRL(0.0,C1DOT)
       CX=C1*1./CF
       VM1=V1-KV1*CR1DOT
       MP1=REALPL(0.0,L1/R1,VM1/R1)
       MT1=KT1*MP1
       MT1E=MT1-BM1*CR1DOT-TL1
       CR1DDT=(1./JTOT1)*MT1E
       CR1DOT=INTGRL(0.0,CR1DDT)
       CR1=INTGRL(0.0,CR1DOT)
```

```
     CRX=CR1*1./CF
     XXDOT=X1DOT*1./CF
     CXDOT=C1DOT*1./CF
     CRXDOT=CR1DOT*1./CF
*****************************************************
KC2DOT=K*C2DOT
     X2DOTE=X2DOT-KC2DOT
     V2=LIMIT(-VSAT,VSAT,K2*X2DOTE)
     C2DDT=V2*KM2
     C2DOT=INTGRL(0.0,C2DDT)
     C2=INTGRL(0.0,C2DOT)
     CY=C2*1./CF
     VM2=V2-KV2*CR2DOT
     MP2=REALPL(0.0,L2/R2,VM2/R2)
     MT2=KT2*MP2
     MT2E=MT2-BM2*CR2DOT-TL2
     CR2DDT=(1./JTOT2)*MT2E
     CR2DOT=INTGRL(0.0,CR2DDT)
     CR2=INTGRL(0.0,CR2DOT)
     CRY=CR2*1./CF
     XYDOT=X2DOT*1./CF
     CYDOT=C2DOT*1./CF
     CRYDOT=CR2DOT*1./CF
*****************************************************
KC3DOT=K*C3DOT
     X3DOTE=X3DOT-KC3DOT
     V3=LIMIT(-VSAT,VSAT,K2*X3DOTE)
     C3DDT=V3*KM3
     C3DOT=INTGRL(0.0,C3DDT)
     C3=INTGRL(0.0,C3DOT)
     CZ=C3*1./CF
     VM3=V3-KV3*CR3DOT
     MP3=REALPL(0.0,L3/R3,VM3/R3)
     MT3=KT3*MP3
     MT3E=MT3-BM3*CR3DOT-TL3
     CR3DDT=(1./JTOT3)*MT3E
     CR3DOT=INTGRL(0.0,CR3DDT)
     CR3=INTGRL(0.0,CR3DOT)
     CRZ=CR3*1./CF
     XZDOT=X3DOT*1./CF
     CZDOT=C3DOT*1./CF
     CRZDOT=CR3DOT*1./CF
***************************************************** SAMPLE
NOSORT
     IF(ABS(V1).LT.VALMIN) GOTO 112
     IF (N1.EQ.0) GOTO 111
     IF (C1DOT.GT.X1DOT) SW1=1
     IF (SW1.EQ.1) GOTO 222
     KS1=ABS(2.*CR1)/(((N1*T)**2)*V1)
     KM1=KS1
```

```
 222          CONTINUE
     C1=CR1
     C1DOT=CR1DOT
 111          N1=N1+1
 112          CONTINUE
*
     IF(ABS(V2).LT.VALMIN) GOTO 334
     IF (N2.EQ.0) GOTO 333
     IF (C2DOT.GT.X2DOT) SW2=1
     IF (SW2.EQ.1) GOTO 444
     KS2=ABS(2.*CR2)/(((N2*T)**2)*V2)
     KM2=KS2
 444          CONTINUE
     C2=CR2
     C2DOT=CR2DOT
 333          N2=N2+1
 334          CONTINUE
*
     IF(ABS(V3).LT.VALMIN) GOTO 556
     IF (N3.EQ.0) GOTO 555
     IF (C3DOT.GT.X3DOT) SW3=1
     IF (SW3.EQ.1) GOTO 666
     KS3=ABS(2.*CR3)/(((N3*T)**2)*V3)
     KM3=KS3
 666          CONTINUE
     C3=CR3
     C3DOT=CR3DOT
 555          N3=N3+1
 556          CONTINUE
SORT
************************************************************ TERMINAL
     WRITE (3,43) RX,RY,RZ,CX,CY,CZ
     CALL CERR(RX,RY,RZ,CX,CY,CZ,DC1,DC2,DC3)
 43           FORMAT(' ',T2,6(F10.8,2X))
     H=H+1
     IF (H.GE.J-1) THEN
                 RETURN
     ENDIF
     CALL NEXPT (RX,RY,RZ,C1,C2,C3)
     CALL CONTIN
METHOD RKSFX
CONTRL FINTIM =10.000,DELT=0.00005 ,DELS=0.00010
PRINT  0.005 ,H,J,RX,RY,RZ,CX,CY,CZ
SAVE   (S1) 0.005,RX,RY,RZ,CX,CY,CZ,DC1,DC2,DC3
GRAPH (G1/S1,DE=TEK618) TIME(LE=8.0,NI=8,UN='SECS'),...
    CX(NI=8,LO=-.5,SC=.25,UN='INCHES')
GRAPH (G2/S1,DE=TEK618) TIME(LE=8.0,NI=8,UN='SECS'),...
    CY(NI=8,LO=-.5,SC=.25,UN='INCHES')
GRAPH (G3/S1,DE=TEK618) TIME(LE=8.0,NI=8,UN='SECS'),...
    CZ(NI=8,LO=-.5,SC=.25,UN='INCHES')
```

```
GRAPH (G4/S2,DE=TEK618) CX(LE=8.0,LO=-.25,NI=8,UN='INCHES'),...
    CY(NI=8,LO=-.25,SC=0.25,UN='INCHES')
GRAPH (G5/S2,DE=TEK618) CX(LE=8.0,LO=-.25,NI=8,UN='INCHES'),...
    CZ(NI=8,LO=-.25,SC=0.25,UN='INCHES')
GRAPH (G6/S2,DE=TEK618) CY(LE=8.0,LO=-.25,NI=8,UN='INCHES'),...
    CZ(NI=8,LO=-.25,SC=0.25,UN='INCHES')
GRAPH (G7/S1,DE=TEK618)  TIME(UN='SECS'),...
     DC1       (UN='ERROR(INCHES)')
GRAPH (G8/S1,DE=TEK618)  TIME(UN='SECS'),...
     DC2       (UN='ERROR(INCHES)')
GRAPH (G9/S1,DE=TEK618)  TIME(UN='SECS'),...
     DC3       (UN='ERROR(INCHES)')
LABEL (G1,G2,G3)   TIME RESPONSE
LABEL (G4,G5,G6)   PLANE PROJECTION
 LABEL (G7,G8,G9)  PATH ERROR
END
STOP
C     FORTRAN SUBROUTINES
FORTRAN
C
C     **********************
C     * SUBROUTINE OPEND *
C     **********************
C
C     TAKE THE VALUE OF J OFF THE TABLE TO BEGIN  SETTING UP FOR
C     INPUT ROUTINE.
C
               SUBROUTINE OPEND (J)
               REWIND 02
               READ (2, *) J
 C  12         FORMAT(I5)
               RETURN
               END
C
C
C     *******************
C     * SUBROUTINE NEXTP *
C     *******************
C
C     INPUTS THE NEXT POINT WHEN THE VALUE OF E DROPS BELOW VALMIN
C     AND CHECKS TO INSURE THAT ALL OF THE VALUES ARE NOT AT ZERO.
C
               SUBROUTINE NEXPT(RX,RY,RZ,C1,C2,C3)
  67           READ (02, *) RX,RY,RZ
               IF ((RX.LE.0.0).AND.(RY.LE.0.0).AND.(RZ.LE.0.0).AND.
     *               (C1.LE.0.0).AND.(C2.LE.0.0).AND.(C3.LE.0.0)) THEN
                     GOTO 67
               ENDIF
               RETURN
               END
```

```
C
C      ******************
C      * SUBROUTINE CERR *
C      ******************
C
C      CHECKS THE ERROR BETWEEN THE DESIRED PATH AND THE SIMULATED
C      RESULT IN THREE DIMENSIONS
C
               SUBROUTINE CERR(RX,RY,RZ,CX,CY,CZ,DC1,DC2,DC3)
               DC1=  (RX-CX)
               DC2=  (RY-CY)
               DC3=  (RZ-CZ)
               RETURN
               END
C
C      **********************************
C      * END ROBOT SIMULATION SUBPROGRAM *
C      **********************************
C
 EOF:
```

```
GRAPH MODIFICATIONS TO ROBSIM2T.FORTRAN USED IN ROBS2T6.FORTRAN.
*   6 PLOTS ON FIRST PAGE-TIME RESPONSE AND PLANE PROJECTIONS
*     3 PATH ERROR ON SECOND PAGE
*     XY PLANE PROJCTION AND 3 PATH ERRORS ON THIRD PAGE
 GRAPH (G1/S1,DE=TEK618,PO=1) TIME(LE=5.0,NI=5,UN='SEC'),...
     CX(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G2/S1,DE=TEK618,OV,PO=1,2.80) TIME(LE=5.0,NI=5,UN='SEC'),...
     CY(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G3/S1,DE=TEK618,OV,PO=1,5.6) TIME(LE=5.0,NI=5,UN='SEC'),...
     CZ(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G4/S1,DE=TEK618,OV,PO=8) CX(LE=2.0,LO=0.0,NI=2,...
      SC=0.5,UN='SEC'), CY(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G5/S1,DE=TEK618,OV,PO=8,2.80) CX(LE=2.0,LO=0.0,NI=2,...
      SC=0.5,UN='SEC'), CZ(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G6/S1,DE=TEK618,OV,PO=8,5.60) CY(LE=2.0,LO=0.0,NI=2,...
      SC=0.5,UN='SEC'), CZ(NI=2,LO=0.0,SC=0.5,UN='INCHES')
 GRAPH (G7/S1,DE=TEK618,PO=1) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC1(NI=2            ,UN='INCHES')
 GRAPH (G8/S1,DE=TEK618,OV,PO=1,2.80) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC2(NI=2            ,UN='INCHES')
 GRAPH (G9/S1,DE=TEK618,OV,PO=1,5.6) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC3(NI=2            ,UN='INCHES')
 GRAPH (G13/S1,DE=TEK618,PO=1,1.75) CX(LE=5.0,LO=0.00,NI=10,SC=.1,...
      UN='INCHES'), CY(NI=10,LO=0.00,LE=5.00,UN='INCHES',SC=.1)
 GRAPH (G14/S1,DE=TEK618,OV,PO=7.5) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC1(NI=2            ,UN='ERR(IN)')
 GRAPH (G15/S1,DE=TEK618,OV,PO=7.5,2.80) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC2(NI=2            ,UN='ERR(IN)')
 GRAPH (G16/S1,DE=TEK618,OV,PO=7.5,5.6) TIME(LE=5.0,NI=5,UN='SEC'),...
      DC3(NI=2            ,UN='ERR(IN)')
 LABEL (G1) TIME RESPONSE
 LABEL (G4) PLANE PROJECTION
 LABEL (G7) PATH ERROR
 LABEL (G10,G11,G12)  PATH ERROR
 LABEL (G13)  XY PLANE PROJECTION
 LABEL (G14) PATH ERROR
```

APPENDIX D
ARTICULATED ROBOT DSL PROGRAMS

This appendix contains the Articulated Robot DSL programs used to model the Articulated Robot Arm. It also contains the results of the single step input used to adjust the step timing for the RAMP Excitations and the Velocity Methods.

RSIMREV1.FORTRAN

This program adapts the Articulated Robot Arm program (Kalogiros, 1987) for multiple run capability, adjusts for zero input, and includes both STEP and RAMP excitation possibilities. Designed for Position Method control it also takes care of 2π to 0.0 movements. It does not adjust for paths whose Cartesian Coordinates pass through (0.5, 0.5, 0.5) which is the Articulated Coordinate System "home" position. Inputted values are a series of three tuples in Articulated Coordinates (CP1,CP2,CP3) which follow a path length integer.

RSIMREV2.FORTRAN

This program is the Velocity version of the proceeding program and was not ran to verify its operation. Modifications are supplied here only for information purposes and are noted in the RSIMREV1 program.

```
TOF:
C       STEVEN G. GOODWAY
C       RSIMREV1.FORTRAN
C       NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C       REVOLUTE ROBOT SIMULATION
C       THIS PROGRAM TAKES THE THREE TUPLE OF ARTICULATED COORDINATES
C       THAT DESCRIBES THE PATH THAT THE ENDOINT OF A ARTICULATED
C       ROBOT IS TO FOLLOW AND APPLIES THE THREE TUPLE TO
C       A DSL PROGRAM FOR EACH ARM AND THE CORRESPONDING ARM
C       MOVEMENT IS COMPAIRED TO THE ORIGINAL THREE TUPLE FOR
C       ACCURACY.
C
 C    FILEDEF'S REQUIRED IN 'DSLGADD EXEC' ARE
 C      FILEDEF 04 DISK DAPATH DATA (RECFM F
 C      FILEDEF 08 DISK APATH4 DATA (RECFM F
C
C       ********************************
C       * SIMULATION FOR ARTICULATED ROBOT *
C       ********************************
C
C       MAIN PROGRAM AREA
C       USER PARAMETER ADJUSTMENTS
C
 PARAM K=1.0,K1=0.6,K2=10000.0,KM1=0.4225,KM2=0.4225,KM3=4.0
 PARAM VSAT=150.0,M1=0.268,M2=0.227,M3=0.041
 PARAM J1=0.033,J2=0.033,J3=0.033,R1=0.91 R2=0.91,R3=0.91
 PARAM KT1=14.4,KT2=14.4,KT3=14.4,L1=0.0001,L2=0.0001,L3=0.0001
 PARAM BM1=0.04297,BM2=0.04297,BM3=0.04297,LOAD=0.0
 PARAM KV1=0.1012,KV2=0.1012,KV3=0.1012,T=0.00025,VALMIN=0.001
 PARAM TV1=0.0075,TV2=0.0075,TV3=0.0075
 PARAM D1=15.,D2=10.,D3=10.,G=386.4
 PARAM REF1=1.,REF2=1.,REF3=1.
 INTEGER SW1,SW2,SW3,N1,N2,N3,J
*
*       K1:             CURVE SCALING CONSTANT
*       K2:             AMPLIFIER GAIN
*       KM:             IDEAL(MODEL) MOTOR CONSTANT
*       VSAT: SATURATION LIMITS OF AMPLIFIER
*       K:              VELOCITY LOOP FEEDBACK GAIN (MODEL)
*       CP1,CP2,CP3:            COMMANDED POSITION IN RADIANS
*       T:              SAMPLING INTERVAL
*       VALMIN:         VALUE OF E USED TO CAUSE THE NEXT STEP TO BE INPUTTED
*       H:              NUMBER OF POINTS ALREADY INPUTTED
*       J:              NUMBER OF POINTS IN PATH (+1)
*
 INITIAL
```

```
J=5
H=0
SW1=0
SW2=0
SW3=0
DC1=0.
DC2=0.
DC3=0.
 N1=0
N2=0
N3=0
C1DT=0.0
C2DT=0.0
C3DT=0.0
C1DDT=0.0
C2DDT=0.0
C3DDT=0.0
X1DOT=0.
X2DOT=0.
X3DOT=0.
CR1=0.
CR2=0.
CR3=0.
CR1DT=0.
CR2DT=0.
CR3DT=0.
CR1DDT=0.
CR2DDT=0.
CR3DDT=0.
TL1=0.
TL2=0.
TL3=0.
MP1=0.
 MP2=0.
MP3=0.
MT1=0.
MT2=0.
MT3=0.
CP1=0.
CP2=0.
CP3=0.
CALL OPEND (J)
CALL FHOME(C1,C2,C3)
CALL NEXPT(H,CP1,CP2,CP3,C1,C2,C3)
      CI1=CP1
      CI2=CP2
```

```
          CI3=CP3
      A1=SQRT(2.*KM1*VSAT)
      A2=SQRT(2.*KM2*VSAT)
      A3=SQRT(2.*KM3*VSAT)
DERIVATIVE
      RR1=CP1*STEP(0.0)
      RR2=CP2*STEP(0.0)
      RR3=CP3*STEP(0.0)
      E1=RR1-C1
      E2=RR2-C2
      E3=RR3-C3
 ******************************************************** NOSORT
 D11  =M3*(D2*COS(CR2)+D3*COS(CR2+CR3))**2+M2*D2**2*(COS(CR2))**2
 D112=2*((M2+M3)*D2**2*COS(CR2)*SIN(CR2)...
    +M3*D3**2*COS(CR2+CR3)*SIN(CR2+CR3)...
    +M3*D2*D3*SIN(2*CR2+CR3))
 D113=2*(M3*D3**2*COS(CR2+CR3)*SIN(CR2+CR3)...
    +M3*D2*D3*COS(CR2)*SIN(CR2+CR3))
 D22 =(M2+M3)*D2**2+M3*D3**2+2*M3*D2*D3*SIN(CR3)
 D23 =M3*D3**2+M3*D2*D3*SIN(CR3)
 D211=(M2+M3)*D2**2*COS(CR2)*SIN(CR2)...
    +M3*D3**2*COS(CR2+CR3)*SIN(CR2+CR3)...
    +M3*D2*D3*SIN(2*CR2+CR3)
 D223=2*M3*D2*D3*SIN(CR3)
 D233= M3*D2*D3*SIN(CR3)
 D32 = M3*D3**2+M3*D2*D3*COS(CR3)
 D33 =M3*D3**2
 D311=M3*D3**2*COS(CR2+CR3)*SIN(CR2+CR3)...
    +M3*D2*D3*COS(CR2)*SIN(CR2+CR3)
 D322=M3*D2*D3*SIN(CR3)
 G2  =(M2+M3)*G*D2*COS(CR2)+M3*G*D3*COS(CR2+CR3)
 G3  =M3*G*D3*COS(CR2+CR3)
 TL1 = -D112*CR1DT*CR2DT-D113*CR1DT*CR3DT
 TL2 = D23*CR3DDT+D211*CR1DT**2-D33*CR3DT**2-D223*CR2DT*CR3DT
 TL3 = D32*CR2DDT+D311*CR1DT**2+D322*CR2DT**2
      JTOT1 = D11+J1
      JTOT2 = D22+J2
      JTOT3 = D33+J3
 ********************************************************

       IF (E1.LT.0.0) X1DOT=-A1*K1*SQRT(ABS(E1))
      IF (E1.GE.0.0) X1DOT= A1*K1*SQRT(E1)
      IF (E2.LT.0.0) X2DOT=-A2*K1*SQRT(ABS(E2))
      IF (E2.GE.0.0) X2DOT= A2*K1*SQRT(E2)
      IF (E3.LT.0.0) X3DOT=-A3*K1*SQRT(ABS(E3))
      IF (E3.GE.0.0) X3DOT= A3*K1*SQRT(E3)
*     USE FOR RAMP OR SINE FUNCTIONS
*     IF ((ABS(C1-CP1).LT.VALMIN).AND.(ABS(C2-CP2).LT.VALMIN) ...
*            AND.(ABS(C3-CP3).LT.VALMIN)) THEN
*   IF ((RR1.GE.CP1).AND.(RR2.GE.CP2).AND.(RR3.GE.CP3))THEN
*            USE FOR STEP FUNCTIONS/ POSITION METHOD
             IF ((ABS(E1).LT.VALMIN).AND.(ABS(E2).LT.VALMIN).AND. ...
                     (ABS(E3).LT.VALMIN)) THEN
             CALL ENDRUN
*                    USE FOR VELOCITY METHOD **********RSIMREV2******
*                    IF (TIME.GE.((H+1)*TSTEP)) THEN
*                    CALL ENDRUN
      ENDIF
*     ENDIF
SORT
*******************************************************
      KC1DOT=K*C1DT
      X1DOTE=X1DOT-KC1DOT
      V1=LIMIT(-VSAT,VSAT,K2*X1DOTE)
      C1DDT=V1*KM1
      C1DT=INTGRL(0.0,C1DDT)
      C1=CI1 +INTGRL(0.0,C1DT)
      VM1=V1-KV1*CR1DT
      MP1=REALPL(0.0,L1/R1,VM1/R1)
      MT1=KT1*MP1
      MT1E=MT1-BM1*CR1DT-TL1
      CR1DDT=(1./JTOT1)*MT1E
      CR1DT=INTGRL(0.0,CR1DDT)
      CR1=INTGRL(0.0,CR1DT)
*******************************************************
      KC2DOT=K*C2DT
      X2DOTE=X2DOT-KC2DOT
      V2=LIMIT(-VSAT,VSAT,K2*X2DOTE)
      C2DDT=V2*KM2
      C2DT=INTGRL(0.0,C2DDT)
      C2=CI2 + INTGRL(0.0,C2DT)
      VM2=V2-KV2*CR2DT
      MP2=REALPL(0.0,L2/R2,VM2/R2)
      MT2=KT2*MP2
      MT2E=MT2-BM2*CR2DT-TL2
      CR2DDT=(1./JTOT2)*MT2E
      CR2DT=INTGRL(0.0,CR2DDT)
      CR2=INTGRL(0.0,CR2DT)
*******************************************************
      KC3DOT=K*C3DT
      X3DOTE=X3DOT-KC3DOT
      V3=LIMIT(-VSAT,VSAT,K2*X3DOTE)
      C3DDT=V3*KM3
      C3DT=INTGRL(0.0,C3DDT)
       C3=CI3 + INTGRL(0.0,C3DT)
      VM3=V3-KV3*CR3DT
       MP3=REALPL(0.0,L3/R3,VM3/R3)
      MT3=KT3*MP3
       MT3E=MT3-BM3*CR3DT-TL3
      CR3DDT=(1./JTOT3)*MT3E
       CR3DT=INTGRL(0.0,CR3DDT)
       CR3=INTGRL(0.0,CR3DT)
************************************************************ SAMPLE
NOSORT
            IF(ABS(V1).LT.VALMIN) GOTO 112
            IF (N1.EQ.0) GOTO 111
            IF (C1DT .GT.X1DOT) SW1=1
            IF (SW1.EQ.1) GOTO 222
            KS1=ABS(2.*CR1)/(((N1*T)**2)*V1)
            KM1=KS1
  222        CONTINUE
            C1=CR1
            C1DT=CR1DT
  111       N1=N1+1
            CONTINUE
  112       CONTINUE
*
            IF(ABS(V2).LT.VALMIN) GOTO 334
            IF (N2.EQ.0) GOTO 333
            IF (C2DT .GT.X2DOT) SW2=1
            IF (SW2.EQ.1) GOTO 444
            KS2=ABS(2.*CR2)/(((N2*T)**2)*V2)
             KM2=KS2
  444       CONTINUE
            C2=CR2
             C2DT=CR2DT
  333       N2=N2+1
            CONTINUE
  334       CONTINUE
*
            IF(ABS(V3).LT.VALMIN) GOTO 556
            IF (N3.EQ.0) GOTO 555
            IF (C3DT .GT.X3DOT) SW3=1
            IF (SW3.EQ.1) GOTO 666
            KS3=ABS(2.*CR3)/(((N3*T)**2)*V3)
             KM3=KS3
  666        CONTINUE
             C3=CR3
            C3DT=CR3DT
```

```
 555        N3=N3+1
            CONTINUE
 556        CONTINUE
SORT
*********************************************************TERMINAL
            CALL CERR (CP1,CP2,CP3,C1,C2,C3,DC1,DC2,DC3)
            WRITE (4,43)  C1,C2,C3
 43         FORMAT(' ',T2,6(F10.8,2X))
            H=H+1
             IF (H.GE.J-1) THEN
                    RETURN
            ENDIF
            CALL NEXPT(H,CP1,CP2,CP3,C1,C2,C3)
            CALL PICRO(CP1,CP2,CP3,C1,C2,C3)
            CALL CONTIN
METHOD RKSFX
CONTRL FINTIM =50.000,DELT=0.00005 ,DELS=0.00010
 PRINT  0.010,H  ,CP1,CP2,CP3,  C1,C2,C3
 SAVE   (S1) 0.005,CP1,CP2,CP3  ,C1,C2,C3,DC1,DC2,DC3
 GRAPH (G1/S1,DE=TEK618,PO=1) TIME(LE=5.0,NI=5,UN='SECS'),...
     C1(NI=2,UN='RADS')
 GRAPH (G2/S1,DE=TEK618,OV,PO=1,2.80) TIME(LE=5.0,NI=5,UN='SECS'),...
     C2(NI=2,UN='RADS')
 GRAPH (G3/S1,DE=TEK618,OV,PO=1,5.6) TIME(LE=5.0,NI=5,UN='SECS'),...
     C3(NI=2,UN='RADS')
 GRAPH (G4/S1,DE=TEK618,PO=1) TIME(LE=5.0,NI=5,UN='SECS'),...
     DC1(NI=2,UN='ERR(RADS)')
 GRAPH (G5/S1,DE=TEK618,OV,PO=1,2.80) TIME(LE=5.0,NI=5,UN='SECS'),...
     DC2(NI=2,UN='ERR(RADS)')
 GRAPH (G6/S1,DE=TEK618,OV,PO=1,5.6) TIME(LE=5.0,NI=5,UN='SECS'),...
     DC3(NI=2,UN='ERR(RADS)')
 LABEL (G1)     TIME RESPONSE
 LABEL (G4) PATH ERROR
 END
 STOP
C      FORTRAN SUBROUTINES
 FORTRAN
C
C      ******************
C      * SUBROUTINE OPEND *
C      ******************
C
            SUBROUTINE OPEND (J)
            REWIND 08
            READ (08, *) J
            REWIND 04
            WRITE(4,*) J
            RETURN
            END
C
C     ******************
C     * SUBROUTINE FHOME *
C     ******************
C
            SUBROUTINE FHOME(C1,C2,C3)
             READ(08,*) C1,C2,C3
            RETURN
            END
C
C     ******************
C     * SUBROUTINE NEXPT *
C     ******************
C
            SUBROUTINE NEXPT(H,CP1,CP2,CP3,C1,C2,C3)
  67        READ (08, *) CP1,CP2,CP3
            IF ((CP1.EQ.C1).AND.(CP2.EQ.C2).AND.(CP3.EQ.C3)) THEN
                   GOTO 67
            ENDIF
            RETURN
            END
C
C     ******************
C     * SUBROUTINE PICRO *
C     ******************
C
C     CHECKS FOR 2*PI CROSSING
            SUBROUTINE PICRO(CP1,CP2,CP3,C1,C2,C3)
            PI=3.141529
            IF ((C1.GT.PI).AND.(CP1.LT.(PI/2))) THEN
                   C1=2*PI-C1
            ENDIF
            RETURN
            END
C
C     *****************
C     * SUBROUTINE CERR *
C     *****************
C
            SUBROUTINE CERR(CP1,CP2,CP3,C1,C2,C3,DC1,DC2,DC3)
            DC1=(CP1-C1)
            DC2=(CP2-C2)
            DC3=(CP3-C3)
            RETURN
            END
C
C           *********************************
C           * END ROBOT SIMULATION SUBPROGRAM *
C           *********************************
C
 EOF:
```

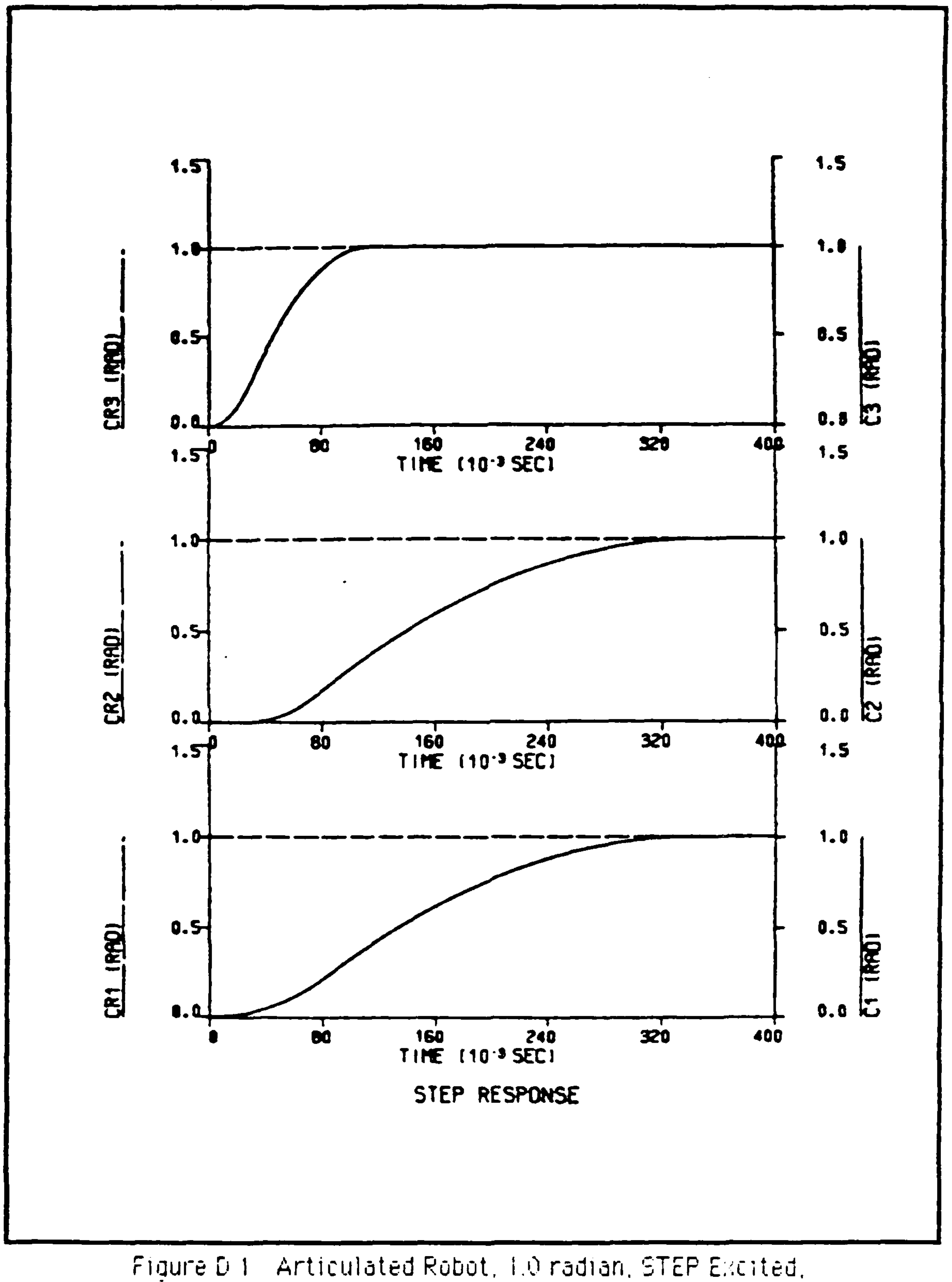


Figure D.1 Articulated Robot, 1.0 radian, STEP Excited, C1, C2, and C3 Step Response.

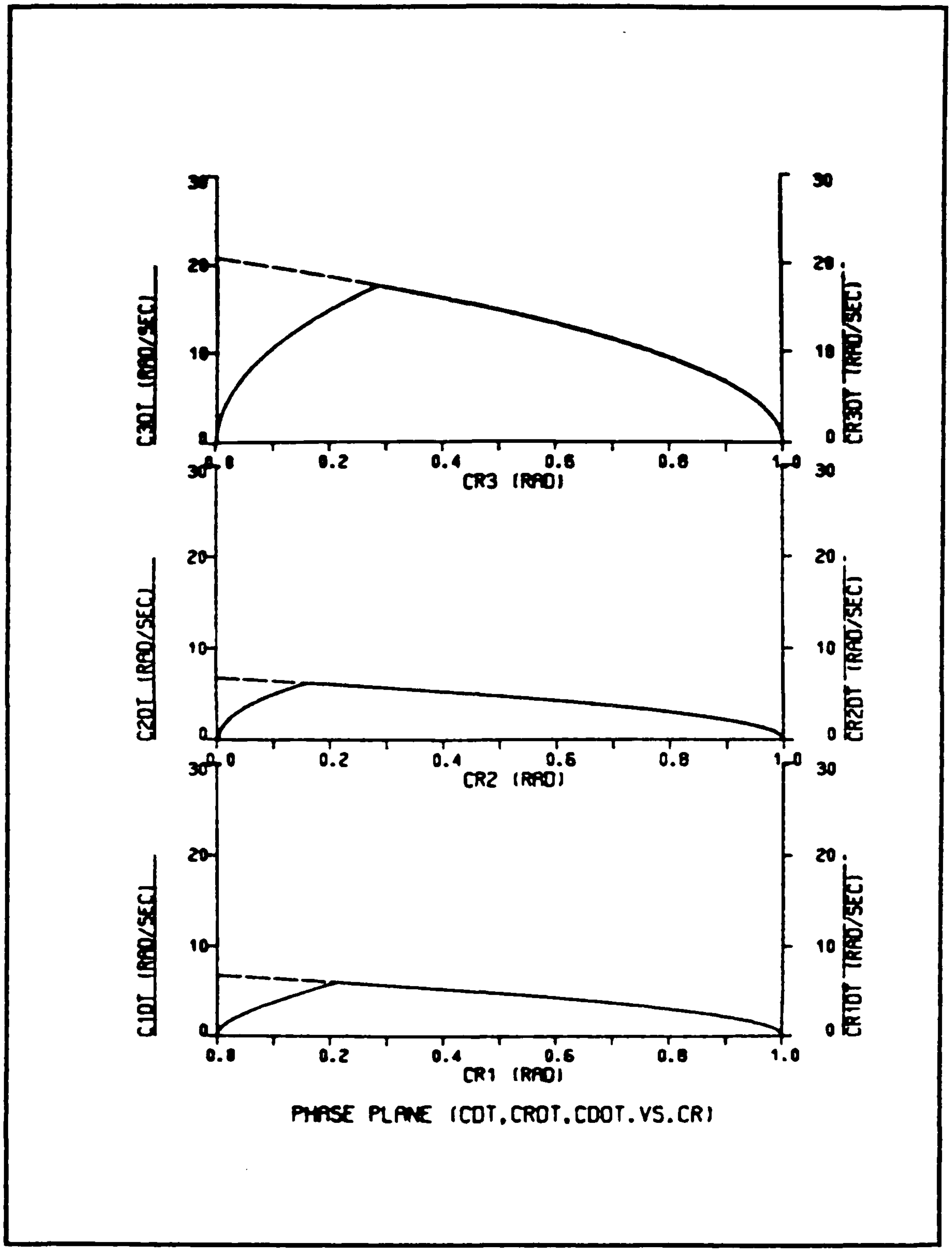


Figure D.2 Articulated Robot, 1.0 radian, STEP Excitation, C1, C2, and C3 Phase Response.

APPENDIX E

PATH CONSTRUCTION PROGRAMS IN FORTRAN

This appendix contains the main path producing program and subroutines. It also contains all of the path modifications for each of the motions conducted in this thesis. The contents of this chapter are:

E.1 ROBOT PATH PROGRAM [ROBOPATH.FORTRAN]

ROBOPATH is the main program from which all of the paths are generated. All of the subroutines are listed at the end of this program.

E.2 LINEAR PATH 1 MAIN PROGRAM

E.3 LINEAR PATH 2 MAIN PROGRAM

E.4 CIRCULAR PATH MAIN PROGRAM

E.5 HELIX PATH 1 AND 2 MAIN PROGRAM

E.6 LINEAR PATH PROGRAM

E.7 SINUSOIDAL PATH MAIN PROGRAM

E.1 ROBOT PATH MAIN PROGRAM

```
 TOF:
C       STEVEN G. GOODWAY
C       ROBOPATH.FORTRAN
C       NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C       VERSION::
C       RPATH3- LINEAR CARTESIAN- MODIFIED FROM VERSION
C       ROBPATH7 FORTRAN--PARAMETRIC SUBROUTINE ADDED
C       CARTESIAN LINEAR MOTION
C       NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C       CARTESIAN ROBOT SIMULATION {ROBSIM2S.FORTRAN}
C       THIS PROGRAM CALCULATES THE THREE TUPLE OF POINTS (X,Y,Z)
C       THAT DESCRIBES THE PATH THAT THE ENDPOINT OF A CARTESIAN
C       ROBOT IS TO FOLLOW.  THEN THE THREE TUPLE IS APPLIED TO
C       A DSL PROGRAM FOR EACH ARM AND THE CORRESPONDING ARM
C       MOVEMENT IS COMPAIRED TO THE ORIGINAL THREE TUPLE FOR
C       ACCURACY.
C
C       ASSUMPTIONS:
C
 C      1. THE MAXIMUM ERROR DURING ANY MOVE IS LESS THAN OR EQUAL
 C       TO THE DISTANCE TRAVELED IN THAT DIMENSION DURING THE
 C      STEP.
 C      2. THE INPUTED EQUATIONS ARE OF THE FORM Y=F(X) AND Z=F(X,Y).
C
C
C       ************************
C       * USER MODIFICATION AREA *
C       ************************
        FUNCTION  EQUAT1(X,Y)
              EQUAT1= X
              RETURN
        END
C       XY PROJECTION FOR EQUAT1
      FUNCTION EQAT1Y(X)
              EQAT1Y=(1.5-X)
              RETURN
      END
 C    EQUATION 2
      FUNCTION  EQUAT2(X,Y)
              EQUAT2=1.0
              RETURN
        END
```

```
C       XY PROJECTION FOR EQUAT2
        FUNCTION EQAT2Y(X)
              EQAT2Y=1.5-X
              RETURN
        END
 C    EQUATION 3
      FUNCTION  EQUAT3(X,Y)
              EQUAT3=1.0
              RETURN
        END
C       XY PROJECTION FOR EQUAT3
        FUNCTION EQAT3Y(X)
              EQAT3Y=1.5-X
              RETURN
        END
 C    EQUATION 4
      FUNCTION  EQUAT4(X,Y)
              EQUAT4=1.0
              RETURN
        END
C       XY PROJECTION FOR EQUAT4
        FUNCTION EQAT4Y(X)
              EQAT4Y=1.5-X
              RETURN
        END
C       PARAMETRIC X EQUATION 1
        FUNCTION PEQX1(T)
              PEQX1= 0.75
              RETURN
        END
C       PARAMETRIC Y EQUATION 1
        FUNCTION PEQY1(T)
              PEQY1= (.5+.25*T)
              RETURN
        END
C       PARAMETRIC Z EQUATION 1
        FUNCTION PEQZ1(T)
              PEQZ1= 0.00
              RETURN
        END
C       PARAMETRIC X EQUATION 2
        FUNCTION PEQX2(T)
              PEQX2= (0.75-0.5*T)
              RETURN
        END
C       PARAMETRIC Y EQUATION 2
```

```
      FUNCTION PEQY2(T)
            PEQY2= 0.75
            RETURN
      END
C     PARAMETRIC Z EQUATION 2
      FUNCTION PEQZ2(T)
            PEQZ2= 0.25
            RETURN
      END
C     PARAMETRIC X EQUATION 3
      FUNCTION PEQX3(T)
            PEQX3= 0.25
            RETURN
      END
C     PARAMETRIC Y EQUATION 3
      FUNCTION PEQY3(T)
            PEQY3= 0.75-.5*T
            RETURN
      END
C     PARAMETRIC Z EQUATION 3
      FUNCTION PEQZ3(T)
            PEQZ3= 0.0
            RETURN
      END
C     PARAMETRIC X EQUATION 4
      FUNCTION PEQX4(T)
            PEQX4= 0.25
            RETURN
      END
C     PARAMETRIC Y EQUATION 4
      FUNCTION PEQY4(T)
            PEQY4=0.25+0.75*T
      RETURN
     END
C     PARAMETRIC Z EQUATION 4
      FUNCTION PEQZ4(T)
            PEQZ4= 1.0
      RETURN
      END
C     END OF FUNCTION DESCRIPTION AREA
C
C     MAIN PROGRAM AREA
C     USER PARAMETER ADJUSTMENTS
      DIMENSION HSPOS(1000,3)
      REAL MAXOD,DELT,MAXODX,MAXODY,MAXODZ,MAXVX,MAXVY,MAXVZ
      COMMON J,I,K,P
      INTEGER N
      DELT=.1
      MAXOD=.05
            PI=3.141529
            ACC=1.0
C     MASS AND INERTIAL ADJUSTMENTS
      MAXVX=1.*MAXOD/DELT
      MAXVY=1.*MAXOD/DELT
      MAXVZ=1.*MAXOD/DELT
      MAXODX=MAXVX*DELT
      MAXODY=MAXVY*DELT
      MAXODZ=MAXVZ*DELT
C     MODIFY DELT FOR PARAMETRIC
      DELT=0.1
C     MIN AND MAX LIMITS
      VALMAX=1E7
      VALMIN=1E-10
C
C     ************************
C     * EQUATION BOUNDS AREA  *
C     ************************
C
C     FIRST PATH EQUATION
C     START/FINISH POINTS
     XS5=0.5
     XF5=1.00
C     CALCULATE Y START/FINISH POINTS
     YS5=EQAT1Y(XS5)
     YF5=EQAT1Y(XF5)
C     CALCULATE Z START/FINISH POINTS
     ZS5=EQUAT1(XS5,YS5)
     ZF5=EQUAT1(XF5,YF5)
C
C     SECOND PATH EQUATION
C     START/FINISH POINTS
C     XS2=0.5
C     XF2=1.0
C     CALCULATE Y START/FINISH POINTS
C     YS2=EQAT2Y(XS2)
C     YF2=EQAT2Y(XF2)
C     CALCULATE Z START/FINISH POINTS
C     ZS2=EQUAT2(XS2,YS2)
C     ZF2=EQUAT2(XF2,YF2)
C
C     PARAMETRIC START AND FINISH POINTS
C     FIRST PARAMETRIC EQUATION
```

```
      TS=0.0
      TF=1.00
      XS1=PEQX1(TS)
      YS1=PEQY1(TS)
      ZS1=PEQZ1(TS)
      XF1=PEQX1(TF)
      YF1=PEQY1(TF)
      ZF1=PEQZ1(TF)
C
C      SECOND PARAMETRIC EQUATION
      XS2=PEQX2(TS)
      YS2=PEQY2(TS)
      ZS2=PEQZ2(TS)
      XF2=PEQX2(TF)
      YF2=PEQY2(TF)
      ZF2=PEQZ2(TF)
C
C      THIRD PARAMETRIC EQUATION
      XS3=PEQX3(TS)
      YS3=PEQY3(TS)
      ZS3=PEQZ3(TS)
      XF3=PEQX3(TF)
      YF3=PEQY3(TF)
      ZF3=PEQZ3(TF)
C
C      FOURTH PARAMETRIC EQUATION
      XS4=PEQX4(TS)
      YS4=PEQY4(TS)
      ZS4=PEQZ4(TS)
      XF4=PEQX4(TF)
      YF4=PEQY4(TF)
      ZF4=PEQZ4(TF)
C
C      OUTPUT START AND FINISH POINTS
C
      WRITE (9,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
      WRITE (6,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
C
      WRITE (9,65) '2',XS2,YS2,ZS2,XF2,YF2,ZF2
      WRITE (6,65) '2',XS2,YS2,ZS2,XF2,YF2,ZF2
C
      WRITE (9,65) '3',XS3,YS3,ZS3,XF3,YF3,ZF3
      WRITE (6,65) '3',XS3,YS3,ZS3,XF3,YF3,ZF3
C
      WRITE (9,65) '4',XS4,YS4,ZS4,XF4,YF4,ZF4
      WRITE (6,65) '4',XS4,YS4,ZS4,XF4,YF4,ZF4
```

```
C
      WRITE (9,65) '5',XS5,YS5,ZS5,XF5,YF5,ZF5
      WRITE (6,65) '5',XS5,YS5,ZS5,XF5,YF5,ZF5
C
C      ************************
C      * CALCULATE PATH MOVEMENT *
C      ************************
C
C
C      HOME
       J=1
       K=1
       HSPOS(1,1)=0.
       HSPOS(1,2)=0.
       HSPOS(1,3)=0.
             WRITE(02,181) HSPOS(1,1),HSPOS(1,2),HSPOS(1,3)
  181        FORMAT(' ',T3,3(F10.8,2X))
C      PATH FORMATION
      XS=XS1
      YS=YS1
      ZS=ZS1
      XF=XF1
      YF=YF1
      ZF=ZF1
C      HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      **********************
C      * START TO FINISH EQUAT1 *
C      **********************
C
C      ADJUST FOR SLOW SPEED
C      MAXODX=MAXODX/ACC
C      MAXODY=MAXODY/ACC
C      MAXODZ=MAXODZ/ACC
       N=1
C      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
C    *         ,J,K)
C
C      START TO FINISH PARAMETRIC EQUATION 1
      CALL SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C      LINEAR FINISH EQUAT1 TO START EQUAT2-FAST
C      MAXODX=ACC*MAXODX
C      MAXODY=ACC*MAXODY
C      MAXODZ=ACC*MAXODZ
      XS=XF1
      YS=YF1
      ZS=ZF1
      XF=XS2
      YF=YS2
      ZF=ZS2
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      ***********************
C      * START TO FINISH EQUAT2 *
C      ***********************
C
C      MAXODX=MAXODX/ACC
C      MAXODY=MAXODY/ACC
C      MAXODZ=MAXODZ/ACC
      XS=XS2
      YS=YS2
      ZS=ZS2
      XF=XF2
      YF=YF2
      ZF=ZF2
      N=2
C      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
C    *           ,J,K)
C      START TO FINISH PARAMETRIC EQUATION 2
      CALL SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C
C      LINEAR FINISH EQUAT2 TO START EQUAT3-FAST
C      MAXODX=ACC*MAXODX
C      MAXODY=ACC*MAXODY
C      MAXODZ=ACC*MAXODZ
      XS=XF2
      YS=YF2
      ZS=ZF2
      XF=XS3
      YF=YS3
      ZF=ZS3
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      ***********************
C      * START TO FINISH EQUAT3 *
C      ***********************
C
C
C      MAXODX=MAXODX/ACC
C      MAXODY=MAXODY/ACC
C      MAXODZ=MAXODZ/ACC
```

```
      XS=XS3
      YS=YS3
      ZS=ZS3
      XF=XF3
      YF=YF3
      ZF=ZF3
      N=3
C      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
C    *           ,J,K)
C      START TO FINISH PARAMETRIC EQUATION 3
      CALL SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C      LINEAR FINISH EQUAT3 TO START EQUAT4-FAST
C      MAXODX=ACC*MAXODX
C      MAXODY=ACC*MAXODY
C      MAXODZ=ACC*MAXODZ
      XS=XF3
      YS=YF3
      ZS=ZF3
      XF=XS4
      YF=YS4
      ZF=ZS4
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      ***********************
C      * START TO FINISH EQUAT4 *
C      ***********************
C
      XS=XF4
      YS=YF4
      ZS=ZF4
      XF=XS4
      YF=YS4
      ZF=ZS4
C      ADJUST FOR SLOW SPEED
C      MAXODX=MAXODX/ACC
C      MAXODY=MAXODY/ACC
C      MAXODZ=MAXODZ/ACC
      N=4
C      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
C    *           ,J,K)
C
C      START TO FINISH PARAMETRIC EQUATION 4
      CALL SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C      LINEAR FINISH EQUAT4 TO START EQUAT5-FAST
C      MAXODX=ACC*MAXODX
C      MAXODY=ACC*MAXODY
```

```
C      MAXODZ=ACC*MAXODZ
      XS=XF4
      YS=YF4
      ZS=ZF4
      XF=XS5
      YF=YS5
      ZF=ZS5
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      **********************
C      * START TO FINISH EQUAT5 *
C      **********************
C
      XS=XS5
      YS=YS5
      ZS=ZS5
      XF=XF5
      YF=YF5
      ZF=ZF5
C      ADJUST FOR SLOW SPEED
C      MAXODX=MAXODX/ACC
C      MAXODY=MAXODY/ACC
C      MAXODZ=MAXODZ/ACC
      N=1
      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
     *        ,J,K)
C
C      START TO FINISH PARAMETRIC EQUATION 5
C      CALL SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C
C      ADJUST FOR FAST MOVEMENT
C      MAXODX=ACC*MAXODX
C      MAXODY=ACC*MAXODY
C      MAXODZ=ACC*MAXODZ
C      FINISH EQUAT5 TO HOME
      XF=XF5
      YF=YF5
      ZF=ZF5
      CALL FHOME(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      **********************
C      * FINISH POINT FORMATION *
C      **********************
C
C
C      PRINTOUT START AND FINISH POINTS
```

```
      WRITE(6,123)
 123       FORMAT('1')
      PRINT *,' '
      PRINT *,'ROBOT SIMULATION 3'
      PRINT *,' '
      PRINT *,'START AND FINISH POINTS'
C     COLUMN HEADERS
      WRITE (9,45) 'NO.','X-START ','Y-START ','Z-START ','X-FINISH',
    *       'Y-FINISH','Z-FINISH'
      WRITE (6,45) 'NO.','X-START ','Y-START ','Z-START ','X-FINISH',
    *       'Y-FINISH','Z-FINISH'
 45        FORMAT (' ',T2,A3,T6,6(A8,4X))
      WRITE (9,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
      WRITE (6,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
 65        FORMAT (' ',T2,A3,6(2X,F10.8 ))
C     READ DATA
      REWIND 02
      DO 555 I=1,J+1
                  READ(2,*) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 555       CONTINUE
C     FORMAT DATA FOR DSL
      REWIND 02
      WRITE (2,*) J+1
      DO 553 I=1,J+1
                  WRITE(2,192) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 553       CONTINUE
 192       FORMAT(' ',T3,3(F10.8,2X))
C     OUTPUT VALUES FOR THREE TUPLE
           PRINT *,' '
      PRINT *,' '
      PRINT *,'THREE TUPLE OF POSITION POINTS'
      WRITE(9,75)'X-POS','Y-POS','Z-POS'
      WRITE(6,75)'X-POS','Y-POS','Z-POS'
 75        FORMAT (' ',T3,3(A5,7X))
      DO 85 I=1,J
                  WRITE(9,95) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
                  WRITE(6,95) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 85        CONTINUE
 95        FORMAT (' ',T2,3(F10.7,2X))
      PRINT *,' '
      PRINT *,'COMPLETED POSITION CALCULATION'
      PRINT *,'COMMENCING ROBOT ARM SIMULATION'
      PRINT *,' '
     END
C
C     ********************
```

```
C      * SUBROUTINE LISTINGS *
C      *********************
C
     SUBROUTINE HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,
    *          MAXODZ,VALMIN,J,K)
C      CALC LINEAR HOME TO START MAXIMIZING STEEPEST PATH
       DIMENSION HSPOS(1000,3)
       REAL MAXODX,MAXODY,MAXODZ,VALMIN
       REWIND 02
       WRITE(9,*) 'HOME TO START'
       WRITE(9,*) XS,YS,ZS
       WRITE(9,*) XF,YF,ZF
C      WRITE(9,*) J,K
       DO 767 I=1,J
            READ (02,*) (HSPOS(I,L),L=1,3)
  767       CONTINUE
       DO 10 J=1,1000
                 DELX=MAXODX
                 DELY=MAXODY
                 DELZ=MAXODZ
C      WRITE (9,*) DELX,DELY,DELZ
C      START POINT IS AT ZERO
       IF ((XS.LT.VALMIN).AND.(YS.LT.VALMIN).AND.(ZS.LT.VALMIN)) THEN
                 GOTO 100
            ENDIF
C
C      SCALE
       IF (ZS.GT.XS) THEN
                 GOTO 20
       ENDIF
       IF (YS.GT.XS) THEN
                 DELX=DELX*(XS/YS)
                 DELZ=DELZ*(ZS/YS)
            ELSE
                 DELY=DELY*(YS/XS)
                 DELZ=DELZ*(ZS/XS)
       ENDIF
       GOTO 30
  20        IF (ZS.GT.YS) THEN
                 DELX=DELX*(XS/ZS)
                 DELY=DELY*(YS/ZS)
            ELSE
                 DELX=DELX*(XS/YS)
                 DELZ=DELZ*(ZS/YS)
       ENDIF
  30        HSPOS(J+1,1)=HSPOS(J,1)+DELX
```

```
        HSPOS(J+1,2)=HSPOS(J,2)+DELY
        HSPOS(J+1,3)=HSPOS(J,3)+DELZ
             WRITE(9,*) HSPOS(J+1,1),HSPOS(J+1,2),HSPOS(J+1,3)
C       CHECK FOR COMPLETION
        IF((HSPOS(J+1,1).GE.XS).AND.(HSPOS(J+1,2).GE.YS).AND.
     *              (HSPOS(J+1,3).GE.ZS)) THEN
                    GOTO 100
        ENDIF
C       CHECK FOR PLANE MOVEMENT
        IF ((HSPOS(J+1,1).GT.XS).AND.(HSPOS(J,1).LT.XS)) THEN
                    HSPOS(J+1,1)=HSPOS(J,1)
        ENDIF
        IF ((HSPOS(J+1,2).GT.YS).AND.(HSPOS(J,2).LT.YS)) THEN
                    HSPOS(J+1,2)=HSPOS(J,2)
        ENDIF
        IF ((HSPOS(J+1,3).GT.ZS).AND.(HSPOS(J,3).LT.ZS)) THEN
                    HSPOS(J+1,3)=HSPOS(J,3)
        ENDIF
 10          CONTINUE
 100         HSPOS(J+1,1)=XS
        HSPOS(J+1,2)=YS
        HSPOS(J+1,3)=ZS
        REWIND 02
        DO 143 I=1,J+1
                    WRITE(2,182) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 143         CONTINUE
 182         FORMAT(' ',T3,3(F10.8,2X))
        RETURN
       END
C
C
        SUBROUTINE STARF(XS,YS,ZS,XF,YF,ZF,N,
     *              MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C       CALCULATE EQUATIONS FROM START TO FINISH
        DIMENSION HSPOS(1000,3)
        REAL MAXODX,MAXODY,MAXODZ,VALMIN
        REWIND 02
        WRITE(9,*) 'START TO FINISH'
        DO 768 I=1,J+1
             READ (02,*) (HSPOS(I,L),L=1,3)
 768         CONTINUE
        DO 11 K=J+1,1000
             DELX=MAXODX
             DELY=MAXODY
             DELZ=MAXODZ
C
```

```
C       CHECK FOR NEGATIVE VALUE DELX,DELY, OR DELZ
        IF ((XF-XS).LT.0.0) THEN
                   DELX=-DELX
        ENDIF
C       ADJUST FOR X PLANE MOVEMENT
        IF (ABS(HSPOS(K,1)-XF).LE.MAXODX) THEN
                   DELX=(ABS(HSPOS(K,1)-XF)*DELX)/(ABS(DELX))
        ENDIF
C       CALCULATE DELY BASED ON DELX OF MAXODX
        IF (N.EQ.1) THEN
              DELY=EQAT1Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
              ELSEIF (N.EQ.2) THEN
              DELY=EQAT2Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
              ELSEIF (N.EQ.3) THEN
              DELY=EQAT3Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
              ELSEIF (N.EQ.4) THEN
              DELY=EQAT4Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
        ENDIF
C       ADJUST FOR PLANE MOVEMENT IN Y PLANE
        IF (ABS(DELY).LE.VALMIN) THEN
                   GOTO 512
        ENDIF
C       ADJUST FOR DELY GREATER THAN MAXODY
       WHILE (ABS(DELY).GT.MAXODY)  DO
              DELX=DELX*(MAXODY/(ABS(DELY)))
C       WRITE (9,*) DELX,DELY,HSPOS(K,1),HSPOS(K,2)
C       ADJUST FOR PLANE MOVEMENT IN X DIRECTION
        IF (ABS(DELX).LE.VALMIN) THEN
                   DELX=0.0
              IF (ABS(HSPOS(K,2)-YF).GT.MAXODY) THEN
                   DELY=(DELY*MAXODY)/(ABS(DELY))
              ELSE
                   DELY=(DELY*ABS(HSPOS(K,2)-YF))/(ABS(DELY))
              ENDIF
                   WRITE (9,*) DELX,DELY
                   GOTO 512
        ENDIF
        IF (N.EQ.1) THEN
                   DELY=EQAT1Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
        ELSEIF (N.EQ.2) THEN
                   DELY=EQAT2Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
        ELSEIF (N.EQ.3) THEN
                   DELY=EQAT3Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
        ELSEIF (N.EQ.4) THEN
                   DELY=EQAT4Y(HSPOS(K,1)+DELX)-HSPOS(K,2)
              ENDIF
```

```
      END WHILE
C     IF ((YF-YS).LT.0) THEN
 C    DELY=-DELY
C     ENDIF
C     CALCULATE DELZ
 512  IF (N.EQ.1) THEN
           DELZ=  (EQUAT1(HSPOS(K,1)+DELX,HSPOS(K,2)+DELY)-HSPOS(K,3))
      ELSEIF (N.EQ.2) THEN
           DELZ=  (EQUAT2(HSPOS(K,1)+DELX,HSPOS(K,2)+DELY)-HSPOS(K,3))
      ELSEIF (N.EQ.3) THEN
           DELZ=  (EQUAT3(HSPOS(K,1)+DELX,HSPOS(K,2)+DELY)-HSPOS(K,3))
      ELSEIF (N.EQ.4) THEN
           DELZ=  (EQUAT4(HSPOS(K,1)+DELX,HSPOS(K,2)+DELY)-HSPOS(K,3))
      ENDIF
C     SCALE FOR MAXIMUM MOVEMENT
           DX=ABS(DELX)
           DY=ABS(DELY)
            DZ=ABS(DELZ)
C     WRITE (9,*) DX,DY,DZ
      IF ((DZ.GT.DX).AND.(DZ.GT.DY)) THEN
           DELX=DELX*MAXODX/DZ
           DELY=DELY*MAXODY/DZ
           DELZ=DELZ*MAXODZ/DZ
      ENDIF
C     WRITE (9,*) '  ',DELX,DELY,DELZ
C     NEXT POINT
      HSPOS(K+1,1)=HSPOS(K,1)+DELX
      HSPOS(K+1,2)=HSPOS(K,2)+DELY
      HSPOS(K+1,3)=HSPOS(K,3)+DELZ
           WRITE(9,183) HSPOS(K  ,1),HSPOS(K  ,2),HSPOS(K  ,3)
C     CHECK FOR PLANE MOVEMENT
      IF (((ABS(HSPOS(K+1,1)-XF)).LT.(ABS(HSPOS(K+1,1)-HSPOS(K,1))))
     *             .AND.(ABS(HSPOS(K,1)-XF).LT.MAXODX)) THEN
                   HSPOS(K+1,1)=XF
           ENDIF
      IF (((ABS(HSPOS(K+1,2)-YF)).LT.(ABS(HSPOS(K+1,2)-HSPOS(K,2))))
     *             .AND.(ABS(HSPOS(K,2)-YF).LT.MAXODY)) THEN
                   HSPOS(K+1,2)=YF
           ENDIF
      IF (((ABS(HSPOS(K+1,3)-ZF)).LT.(ABS(HSPOS(K+1,3)-HSPOS(K,3))))
     *             .AND.(ABS(HSPOS(K,3)-ZF).LT.MAXODZ)) THEN
                   HSPOS(K+1,3)=ZF
           ENDIF
C     CHECK FOR COMPLETION
      IF ((ABS(HSPOS(K,1)-XF).LT.MAXODX).AND.(ABS(HSPOS(K,2)-YF).LT.
     *             MAXODY).AND.(ABS(HSPOS(K,3)-ZF).LT.MAXODZ)) THEN
```

```
                 GOTO 101
      ENDIF
 11        CONTINUE
C     SAVE DATA
 101       REWIND 02
           HSPOS(K+1,1)=XF
           HSPOS(K+1,2)=YF
           HSPOS(K+1,3)=ZF
      DO 243 I= 1,K+1
                 WRITE(2,183) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 243       CONTINUE
 183       FORMAT(' ',T3,3(F10.8,2X))
      RETURN
      END
C
C
      SUBROUTINE FSTAR(XS,YS,ZS,XF,YF,ZF,
     *           MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     CALCULATES LINEAR FINISH TO NEXT START
      DIMENSION HSPOS(1000,3)
      REAL MAXODX,MAXODY,MAXODZ,VALMIN
      REWIND 02
      WRITE(9,*) 'FINISH TO START'
      WRITE(9,*) XS,YS,ZS
      WRITE(9,*) XF,YF,ZF
C          WRITE(9,*) J,K+1
      DO 769 I=1,K+1
      READ (02,*) (HSPOS(I,L),L=1,3)
 769       CONTINUE
      DO 12 J=K+1, 1000
C     ALL START AND FINISH POINTS THE SAME
      IF ((XS.EQ.XF).AND.(YS.EQ.YF).AND.(ZS.EQ.ZF)) THEN
                 GOTO 102
      ENDIF
      DELX=MAXODX
      DELY=MAXODY
      DELZ=MAXODZ
      XFS= ABS(XS-XF)
      YFS=ABS(YS-YF)
      ZFS=ABS(ZS-ZF)
C
      IF (ZFS.GE.XFS) THEN
                 GOTO 22
      ENDIF
      IF (YFS.GE.XFS) THEN
                 DELX=DELX*(XFS/YFS)
```

```
                    DELZ=DELZ*(ZFS/YFS)
            ELSEIF (XFS.GE.YFS) THEN
                    DELY=DELY*(YFS/XFS)
                    DELZ=DELZ*(ZFS/XFS)
        ENDIF
        GOTO 32
 22         IF (ZFS.GE.YFS) THEN
                    DELX=DELX*(XFS/ZFS)
                    DELY=DELY*(YFS/ZFS)
            ELSEIF (YFS.GE.ZFS) THEN
                    DELX=DELX*(XFS/YFS)
                    DELZ=DELZ*(ZFS/YFS)
        ENDIF
C       CHECK FOR NEGATIVE DELX , DELY , OR DELZ
 32         IF ((XF-XS).LT.0) THEN
                    DELX=-ABS(DELX)
        ENDIF
        IF ((YF-YS).LT.0) THEN
                    DELY=-ABS(DELY)
        ENDIF
        IF ((ZF-ZS).LT.0) THEN
                    DELZ=-ABS(DELZ)
        ENDIF
C       NEXT POINT
        HSPOS(J+1,1)=HSPOS(J,1)+DELX
        HSPOS(J+1,2)=HSPOS(J,2)+DELY
        HSPOS(J+1,3)=HSPOS(J,3)+DELZ
        WRITE(9,184)HSPOS(J+1,1),HSPOS(J+1,2),HSPOS(J+1,3)
C       CHECK FOR PLANE MOVEMENT
        IF (((ABS(HSPOS(J+1,1)-XF)).LT.(ABS(HSPOS(J+1,1)-HSPOS(J,1))))
     *              .AND.(ABS(HSPOS(J,1)-XF).LT.MAXODX)) THEN
                    HSPOS(J+1,1)=XF
            ENDIF
        IF (((ABS(HSPOS(J+1,2)-YF)).LT.(ABS(HSPOS(J+1,2)-HSPOS(J,2))))
     *              .AND.(ABS(HSPOS(J,2)-YF).LT.MAXODY)) THEN
                    HSPOS(J+1,2)=YF
            ENDIF
        IF (((ABS(HSPOS(J+1,3)-ZF)).LT.(ABS(HSPOS(J+1,3)-HSPOS(J,3))))
     *              .AND.(ABS(HSPOS(J,3)-ZF).LT.MAXODZ)) THEN
                    HSPOS(J+1,3)=ZF
            ENDIF
C       CHECK FOR COMPLETION
        IF ((ABS(HSPOS(J,1)-XF).LE.ABS(DELX)).AND.(ABS(HSPOS(J,2)-YF)
     *              .LE.ABS(DELY)).AND.(ABS(HSPOS(J,3)-ZF).LE.ABS(DELZ))) THEN
                    GOTO 102
        ENDIF
 12          CONTINUE
 102         HSPOS(J+1,1)=XF
      HSPOS(J+1,2)=YF
      HSPOS(J+1,3)=ZF
C     SAVE DATA
      REWIND 02
      DO 343 I=  1,J+1
                      WRITE(2,184) HSPOS(I,1),HSPOS(I,2),HSPCS(I,3)
C                     WRITE(9,184) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 343         CONTINUE
 184         FORMAT(' ',T3,6(F10.8,2X))
      RETURN
      END
C
      SUBROUTINE  FHOME(XS,YS,ZS,XF,YF,ZF,
     * MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     CALC LINEAR FINISH TO HOME MAXIMIZING STEEPEST PATH
      DIMENSION HSPOS(1000,3)
      REAL MAXODX,MAXODY,MAXODZ,VALMIN
      REWIND 02
      WRITE(9,*) 'FINISH TO HOME '
      DO 766 I=1,K+1
             READ (02,*) (HSPOS(I,L),L=1,3)
 766         CONTINUE
      DO 13 J=K+1,1000
                      DELX=MAXODX
                      DELY=MAXODY
                      DELZ=MAXODZ
C     FINISH POINT IS AT ZERO
      IF ((XF.LT.VALMIN).AND.(YF.LT.VALMIN).AND.(ZF.LT.VALMIN)) THEN
                      GOTO 103
             ENDIF
C
      IF (ZF.GE.XF) THEN
                      GOTO 23
      ENDIF
      IF (YF.GE.XF) THEN
                      DELX=DELX*(XF/YF)
                      DELZ=DELZ*(ZF/YF)
             ELSE
                      DELY=DELY*(YF/XF)
                      DELZ=DELZ*(ZF/XF)
      ENDIF
      GOTO 33
 23          IF (ZF.GE.YF) THEN
                      DELX=DELX*(XF/ZF)
```

```
                     DELY=DELY*(YF/ZF)
            ELSE
                     DELX=DELX*(XF/YF)
                     DELZ=DELZ*(ZF/YF)
     ENDIF
  33          HSPOS(J+1,1)=HSPOS(J,1)-DELX
     HSPOS(J+1,2)=HSPOS(J,2)-DELY
     HSPOS(J+1,3)=HSPOS(J,3)-DELZ
              WRITE(9,185) HSPOS(J+1,1),HSPOS(J+1,2),HSPOS(J+1,3)
C    CHECK FOR COMPLETION
     IF((HSPOS(J+1,1).LE.0.).AND.(HSPOS(J+1,2).LE.0.).AND.
   *                 (HSPOS(J+1,3).LE.0.)) THEN
                     GOTO 103
     ENDIF
C    CHECK FOR PLANE MOVEMENT
     IF ((HSPOS(J+1,1).LT.XF).AND.(HSPOS(J,1).GT.XF)) THEN
                     HSPOS(J+1,1)=HSPOS(J,1)
     ENDIF
     IF ((HSPOS(J+1,2).LT.YF).AND.(HSPOS(J,2).GT.YF)) THEN
                     HSPOS(J+1,2)=HSPOS(J,2)
     ENDIF
     IF ((HSPOS(J+1,3).LT.ZF).AND.(HSPOS(J,3).GT.ZF)) THEN
                     HSPOS(J+1,3)=HSPOS(J,3)
     ENDIF
  13          CONTINUE
  103         HSPOS(J+1,1)=0.
     HSPOS(J+1,2)=0.
     HSPOS(J+1,3)=0.
C    SAVE DATA
    REWIND 02
    DO 443 I=  1,J+1
      WRITE(2,185) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 C  WRITE(9,185) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
  443  CONTINUE
  185  FORMAT(' ',T3,3(F10.8,2X))
    RETURN
    END
C    PARAMETRIC SUBROUTINE SFPAR
    SUBROUTINE SFPAR(N,TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
     DIMENSION HSPOS(1000,3)
    REAL MAXODX,MAXODY,MAXODZ,VALMIN,DELT,TS,TF
    DT=DELT
     REWIND 02
      DO 810 I=1,J+1
           READ (2,*)(HSPOS(I,L),L=1,3)
  810 CONTINUE
```

```
        T=TS
                K=J+1
        WHILE (T.LT.TF) DO
820             IF (N.EQ.1) THEN
                        DELX=PEQX1(T+DT)-PEQX1(T)
                ELSEIF (N.EQ.2) THEN
                        DELX=PEQX2(T+DT)-PEQX2(T)
                ELSEIF (N.EQ.3) THEN
                        DELX=PEQX3(T+DT)-PEQX3(T)
                ELSEIF (N.EQ.4) THEN
                        DELX=PEQX4(T+DT)-PEQX4(T)
        ENDIF
                IF (ABS(DELX).GT.MAXODX) THEN
                        DT=ABS(DT*MAXODX/DELX)
                        GOTO 820
                ELSEIF ((ABS(DELX).LE.VALMIN).OR.(DT.LE.VALMIN))THEN
                        DELX=0.0
                        DT=DELT
                ENDIF
                DTT=DT
830             IF (N.EQ.1) THEN
                        DELY=PEQY1(T+DT)-PEQY1(T)
                ELSEIF (N.EQ.2) THEN
                        DELY=PEQY2(T+DT)-PEQY2(T)
                ELSEIF (N.EQ.3) THEN
                        DELY=PEQY3(T+DT)-PEQY3(T)
                ELSEIF (N.EQ.4) THEN
                        DELY=PEQY4(T+DT)-PEQY4(T)
        ENDIF
                IF (ABS(DELY).GT.MAXODY) THEN
                        DT=ABS(DT*MAXODY/DELY)
                        GOTO 830
                ELSEIF ((ABS(DELY).LE.VALMIN).OR.(DT.LE.VALMIN))THEN
                        DELY=0.0
                        DT=DTT
                ENDIF
                DTT=DT
840             IF (N.EQ.1) THEN
                        DELZ=PEQZ1(T+DT)-PEQZ1(T)
                ELSEIF (N.EQ.2) THEN
                        DELZ=PEQZ2(T+DT)-PEQZ2(T)
                ELSEIF (N.EQ.3) THEN
                        DELZ=PEQZ3(T+DT)-PEQZ3(T)
                ELSEIF (N.EQ.4) THEN
                        DELZ=PEQZ4(T+DT)-PEQZ4(T)
        ENDIF
```

```
            IF (ABS(DELZ).GT.MAXODZ) THEN
                    DT=ABS(DT*MAXODZ/DELZ)
                    GOTO 840
            ELSEIF ((ABS(DELZ).LE.VALMIN).OR.(DT.LE.VALMIN))THEN
                    DELZ=0.0
                    DT=DTT
            ENDIF
C     MAXED OUT YET
      IF ((T+DT).GT.TF) THEN
                    DT=TF-T
      ENDIF
C   CALCULATE NEXT POINT
            IF (N.EQ.1) THEN
                    DELX=PEQX1(T+DT)-PEQX1(T)
                    DELY=PEQY1(T+DT)-PEQY1(T)
                    DELZ=PEQZ1(T+DT)-PEQZ1(T)
            ELSEIF (N.EQ.2) THEN
                    DELX=PEQX2(T+DT)-PEQX2(T)
                    DELY=PEQY2(T+DT)-PEQY2(T)
                    DELZ=PEQZ2(T+DT)-PEQZ2(T)
            ELSEIF (N.EQ.3) THEN
                     DELX=PEQX3(T+DT)-PEQX3(T)
                    DELY=PEQY3(T+DT)-PEQY3(T)
                    DELZ=PEQZ3(T+DT)-PEQZ3(T)
            ELSEIF (N.EQ.4) THEN
                    DELX=PEQX4(T+DT)-PEQX4(T)
                    DELY=PEQY4(T+DT)-PEQY4(T)
                    DELZ=PEQZ4(T+DT)-PEQZ4(T)
      ENDIF
C  WRITE (9,115) DT,DELX,DELY,DELZ
 115        FORMAT (' ',T3,4(F6.4,3X))
      HSPOS(K+1,1)=HSPOS(K,1)+DELX
      HSPOS(K+1,2)=HSPOS(K,2)+DELY
      HSPOS(K+1,3)=HSPOS(K,3)+DELZ
            WRITE(9,183) HSPOS(K,1),HSPOS(K,2),HSPOS(K,3)
      T=T+DT
      K=K+1
      END WHILE
C     SAVE RESULTS
     REWIND 02
      DO 243 I=  1,K
                    WRITE(2,183) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
C                   WRITE(9,183) HSPOS(I,1),HSPOS(I,2),HSPOS(I,3)
 243        CONTINUE
      K=K-1
 183        FORMAT(' ',T3,3(F10.8,2X))
```

```
          RETURN
          END
      EOF:
```

E.2 LINEAR PATH1 MODIFICATIONS

```
TOF:
C       STEVEN G. GOODWAY
C       RPATH2- LINEAR CARTESIAN- MODIFIED FROM VERSION
C
C       ************************
C       * USER MODIFICATION AREA *
C       ************************
        FUNCTION  EQUAT1(X,Y)
              EQUAT1= .500
              RETURN
        END
C       XY PROJECTION FOR EQUAT1
        FUNCTION EQAT1Y(X)
              EQAT1Y=1.25-X
              RETURN
        END
C       EQUATION 2
        FUNCTION  EQUAT2(X,Y)
              EQUAT2=1.0
              RETURN
        END
C       XY PROJECTION FOR EQUAT2
        FUNCTION EQAT2Y(X)
              EQAT2Y=1.5-X
              RETURN
        END
C       EQUATION 3
        FUNCTION  EQUAT3(X,Y)
              EQUAT3=1.0
              RETURN
        END
C       XY PROJECTION FOR EQUAT3
        FUNCTION EQAT3Y(X)
              EQAT3Y=1.5-X
              RETURN
        END
C       EQUATION 4
        FUNCTION  EQUAT4(X,Y)
              EQUAT4=1.0
              RETURN
        END
C       XY PROJECTION FOR EQUAT4
        FUNCTION EQAT4Y(X)
              EQAT4Y=1.5-X
```

```
            RETURN
        END
C       PARAMETRIC X
        FUNCTION PEQX(T)
            PEQX= .5*COS(T)+.5
        RETURN
        END
C       PARAMETRIC Y
        FUNCTION PEQY(T)
            PEQY= .5*SIN(T)+.5
        RETURN
        END
C       PARAMETRIC Z
        FUNCTION PEQZ(T)
            PEQZ= T/(12*3.141529)
        RETURN
        END
C       END OF FUNCTION DESCRIPTION AREA
C
C       ***********************
C       * EQUATION BOUNDS AREA  *
C       ***********************
C       FIRST PATH EQUATION
C       START/FINISH POINTS
        XS1=1.
        XF1=0.25
C       CALCULATE Y START/FINISH POINTS
        YS1=EQAT1Y(XS1)
        YF1=EQAT1Y(XF1)
C       CALCULATE Z START/FINISH POINTS
        ZS1=EQUAT1(XS1,YS1)
        ZF1=EQUAT1(XF1,YF1)
C       OUTPUT START AND FINISH POINTS FOR EQUATION 1
        WRITE (9,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
        WRITE (6,65) '1',XS1,YS1,ZS1,XF1,YF1,ZF1
C       SECOND PATH EQUATION
C       START/FINISH POINTS
        XS2=0.5
        XF2=1.0
C       CALCULATE Y START/FINISH POINTS
        YS2=EQAT2Y(XS2)
        YF2=EQAT2Y(XF2)
C       CALCULATE Z START/FINISH POINTS
        ZS2=EQUAT2(XS2,YS2)
        ZF2=EQUAT2(XF2,YF2)
C       OUTPUT START AND FINISH POINTS FOR EQUATION 2
```

```
      WRITE (9,65) '2',XS2,YS2,ZS2,XF2,YF2,ZF2
      WRITE (6,65) '2',XS2,YS2,ZS2,XF2,YF2,ZF2
C
C**************************
C* CALCULATE PATH MOVEMENT *
C**************************
C     PATH FORMATION
      XS=XS1
      YS=YS1
      ZS=ZS1
      XF=XF1
      YF=YF1
      ZF=ZF1
C     HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C     ***********************
C     * START TO FINISH EQUAT1 *
C     ***********************
      N=1
      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
     *      ,J,K)
C     LINEAR FINISH EQUAT1 TO START EQUAT2-FAST
      XS=XF1
      YS=YF1
      ZS=ZF1
      XF=XS2
      YF=YS2
      ZF=ZS2
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     START TO FINISH EQUATION 2
            XS=XS2
      YS=YS2
      ZS=ZS2
      XF=XF2
      YF=YF2
      ZF=ZF2
      N=2
      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
     *      ,J,K)
C     FINISH EQUAT2 TO HOME
      XF=XF2
      YF=YF2
      ZF=ZF2
      CALL FHOME(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
```

```
C       ***********************
C       * FINISH POINT FORMATION *
C       ***********************
```

END MODIFICATION REGION

E.3 LINEAR PATH 2 PROGRAM

DESCRIBED IN SECTION E.1

E.4 CIRCULAR PATH MOTION

```
TOF:
C      STEVEN G. GOODWAY
C      ROBPATH4 FORTRAN
C
C      ****************************
C      * USER MODIFICATION AREA *
C      ****************************
       FUNCTION  EQUAT1(X,Y)
                EQUAT1=0.5
                RETURN
       END
C      XY PROJECTION FOR EQUAT1
       FUNCTION EQAT1Y(X)
                EQAT1Y=(SQRT(ABS(.25-(X-.5)**2)))+.5
                RETURN
       END
       FUNCTION  EQUAT2(X,Y)
                EQUAT2=0.5
                RETURN
       END
C      XY PROJECTION FOR EQUAT2
       FUNCTION EQAT2Y(X)
                EQAT2Y=-(SQRT(ABS(.25-(X-.5)**2)))+.5
                RETURN
       END
C      END OF FUNCTION DESCRIPTION AREA
C
C      MAIN PROGRAM AREA
C      USER PARAMETER ADJUSTMENTS
C
C      ***********************
C      * EQUATION BOUNDS AREA *
C      ***********************
C      FIRST PATH EQUATION
C      START/FINISH POINTS
       XS1=1.
       YS1=0.5
       XF1=0.0
       YF1=0.5
C      CALCULATE Z START/FINISH POINTS
       ZS1=EQUAT1(XS1,YS1)
       ZF1=EQUAT1(XF1,YF1)
C
C               SECOND PATH EQUATION
```

```
C     START/FINISH POINTS
      XS2=0.0
      YS2=0.5
      XF2=1.0
      YF2=0.5
C     CALCULATE Z START/FINISH POINTS
      ZS2=EQUAT2(XS2,YS2)
      ZF2=EQUAT2(XF2,YF2)
C
C     ***************************
C     * CALCULATE PATH MOVEMENT *
C     ***************************
C     PATH FORMATION
      XS=XS1
      YS=YS1
      ZS=ZS1
      XF=XF1
      YF=YF1
      ZF=ZF1
C     HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C     ***********************
C     * START TO FINISH EQUAT1 *
C     ***********************
      ACC=0.01
C     ADJUST FOR SLOW SPEED
      MAXODX=MAXODX/ACC
      MAXODY=MAXODY/ACC
      MAXODZ=MAXODZ/ACC
      N=1
     CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
    *  ,J,K)
C
C     LINEAR FINISH EQUAT1 TO START EQUAT2-FAST
      MAXODX=ACC*MAXODX
      MAXODY=ACC*MAXODY
      MAXODZ=ACC*MAXODZ
      XS=XF1
      YS=YF1
      ZS=ZF1
      XF=XS2
      YF=YS2
      ZF=ZS2
     CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     START TO FINISH EQUATION 2
```

```
C
      MAXODX=MAXODX/ACC
      MAXODY=MAXODY/ACC
                MAXODZ=MAXODZ/ACC
      XS=XS2
      YS=YS2
      ZS=ZS2
      XF=XF2
      YF=YF2
      ZF=ZF2
      N=2
     CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ.VALMIN
    * ,J,K)
C
C     ADJUST FOR FAST MOVEMENT
      MAXODX=ACC*MAXODX
      MAXODY=ACC*MAXODY
      MAXODZ=ACC*MAXODZ
C     FINISH EQUAT3 TO HOME
      XF=XF2
      YF=YF2
      ZF=ZF2
      CALL FHOME(XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C     **********************
C     * FINISH POINT FORMATION *
C     **********************
```

E.5 HELIX PATH 1 AND 2

```
TOF:
C       STEVEN G. GOODWAY
C       ROBPATH5 FORTRAN--PARAMETRIC SUBROUTINE ADDED
C       NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C       ************************
C       * USER MODIFICATION AREA *
C       ************************
      FUNCTION  EQUAT1(X,Y)
              EQUAT1=0.5
              RETURN
       END
C       XY PROJECTION FOR EQUAT1
      FUNCTION EQAT1Y(X)
              EQAT1Y=(SQRT(ABS(.25-(X-.5)**2)))+.5
              RETURN
       END
       FUNCTION  EQUAT2(X,Y)
 C     EQUAT2=((ABS(.25-(X-.5)**2 -(Y-.5)**2 ))**.5+.5)
              EQUAT2=0.5
              RETURN
       END
C       XY PROJECTION FOR EQUAT2
       FUNCTION EQAT2Y(X)
              EQAT2Y=-(SQRT(ABS(.25-(X-.5)**2)))+.5
              RETURN
       END
C       PARAMETRIC X
       FUNCTION PEQX(T)
              PEQX= .5*COS(T)+.5
       RETURN
       END
C       PARAMETRIC Y
       FUNCTION PEQY(T)
              PEQY= .5*SIN(T)+.5
       RETURN
       END
C       PARAMETRIC Z
       FUNCTION PEQZ(T)
C       HELIX 3-CYCLE
C             PEQZ=T/(6*3.141529)
C       HELIX 6-CYCLE
              PEQZ= T/(12*3.141529)
       RETURN
       END
```

```
C     END OF FUNCTION DESCRIPTION AREA
C
C     MAIN PROGRAM AREA
C     USER PARAMETER ADJUSTMENTS
            ACC=1.0
C     MODIFY DELT FOR PARAMETRIC
     DELT=0.1
C     ***********************
C     * EQUATION BOUNDS AREA  *
C     ***********************
C     FIRST PATH EQUATION
C     PARAMETRIC START AND FINISH POINTS
      TS=0.0
C     HELIX 3-CYCLE
C               TF=6*PI
C     HELIX 6-CYCLE
      TF=12*PI
      XS1=PEQX(TS)
      YS1=PEQY(TS)
      ZS1=PEQZ(TS)
      XF1=PEQX(TF)
      YF1=PEQY(TF)
      ZF1=PEQZ(TF)
C
C     *************************
C     * CALCULATE PATH MOVEMENT *
C     *************************
C     PATH FORMATION
      XS=XS1
      YS=YS1
      ZS=ZS1
      XF=XF1
      YF=YF1
      ZF=ZF1
C     HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     ***********************
C     * START TO FINISH EQUAT1 *
C     ***********************
C     START TO FINISH PARAMETRIC EQUATION
     CALL SFPAR(TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN)
C     FINISH EQUAT3 TO HOME
      XF=XF1
      YF=YF1
      ZF=ZF1
     CALL FHOME(XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
```

```
C
C     ************************
C     * FINISH POINT FORMATION *
C     ************************
C

END HELIX PATH 1 & 2 MODIFICATION
```

E.6 LINEAR PATH MODIFICATIONS

```
TOF:
C      STEVEN G. GOODWAY
C      ROBPATH6 FORTRAN--PARAMETRIC SUBROUTINE ADDED
C      LINEAR PARAMETRIC/CATRSIAN
C      ************************
C      * USER MODIFICATION AREA *
C      ************************
C      PARAMETRIC X --EQUATION 1
       FUNCTION PEQX1(T)
              PEQX1= 1.
       RETURN
       END
C      PARAMETRIC Y --EQUATION 1
       FUNCTION PEQY1(T)
              PEQY1= .25+T
       RETURN
       END
C      PARAMETRIC Z  --EQUATION 1
       FUNCTION PEQZ1(T)
              PEQZ1= .5
       RETURN
       END
C      PARAMETRIC X --EQUATION 2
       FUNCTION PEQX2(T)
              PEQX2= .75-T
       RETURN
       END
C      PARAMETRIC Y --EQUATION 2
       FUNCTION PEQY2(T)
              PEQY2= 1.
       RETURN
       END
C      PARAMETRIC Z  --EQUATION 2
       FUNCTION PEQZ2(T)
              PEQZ2= .5
       RETURN
       END
C      PARAMETRIC X --EQUATION 3
       FUNCTION PEQX3(T)
              PEQX3= 0.0
       RETURN
       END
C      PARAMETRIC Y --EQUATION 3
       FUNCTION PEQY3(T)
```

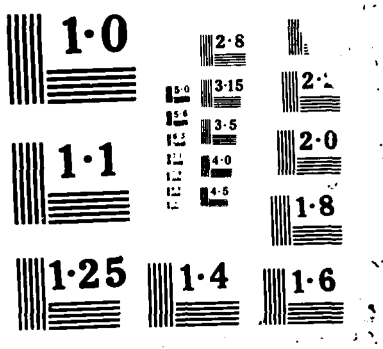
```
              PEQY3= 1.
        RETURN
        END
C       PARAMETRIC Z --EQUATION 3
        FUNCTION PEQZ3(T)
              PEQZ3= 2.*T
        RETURN
        END
C       CARTESIAN --EQUATION 4
        FUNCTION  EQUAT4(X,Y)
              EQUAT4=Y
              RETURN
        END
C       XY PROJECTION FOR EQUAT4
        FUNCTION EQAT4Y(X)
              EQAT4Y=1. - X
              RETURN
        END
C       END OF FUNCTION DESCRIPTION AREA
C       MAIN PROGRAM AREA
C       USER PARAMETER ADJUSTMENTS
C       ************************
C       * EQUATION BOUNDS AREA  *
C       ************************
C
C       PARAMETRIC START AND FINISH POINTS
        TS=0.0
        TF=0.5
C       PARAMETRIC EQUATION 1 START/FINISH
       XS1=PEQX1(TS)
       YS1=PEQY1(TS)
       ZS1=PEQZ1(TS)
       XF1=PEQX1(TF)
       YF1=PEQY1(TF)
       ZF1=PEQZ1(TF)
C       PARAMETRIC EQUATION 2 START/FINISH
       XS2=PEQX2(TS)
       YS2=PEQY2(TS)
       ZS2=PEQZ2(TS)
       XF2=PEQX2(TF)
       YF2=PEQY2(TF)
       ZF2=PEQZ2(TF)
C       PARAMETRIC EQUATION 3 START/FINISH
       XS3=PEQX3(TS)
       YS3=PEQY3(TS)
       ZS3=PEQZ3(TS)
```

```
      XF3=PEQX3(TF)
      YF3=PEQY3(TF)
      ZF3=PEQZ3(TF)
C      CARTESIAN EQUATION 4  START/FINISH
      XS4=0.75
      XF4=0.25
C      CALCULATE Y START/FINISH POINTS
      YS4=EQAT4Y(XS4)
      YF4=EQAT4Y(XF4)
C      CALCULATE Z START/FINISH POINTS
      ZS4=EQUAT4(XS4,YS4)
      ZF4=EQUAT4(XF4,YF4)
C
C
```

```
C      ************************
C      * CALCULATE PATH MOVEMENT *
C      ************************
C
C      PATH FORMATION
      XS=XS1
      YS=YS1
      ZS=ZS1
      XF=XF1
      YF=YF1
      ZF=ZF1
C      HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C      **********************
C      * START TO FINISH EQUAT1 *
C      **********************
C      START TO FINISH PARAMETRIC EQUATION
      N=1
      CALL SFPAR(TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN,N)
C      LINEAR FINISH EQUAT1 TO START EQUAT2
      XS=XF1
      YS=YF1
      ZS=ZF1
      XF=XS2
      YF=YS2
      ZF=ZS2
      WRITE (9,*) XS,YS,ZS,XF,YF,ZF
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C      START TO FINISH EQUATION 2
      XS=XS2
      YS=YS2
      ZS=ZS2
      XF=XF2
      YF=YF2
      ZF=ZF2
      N=2
      CALL SFPAR(TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN,N)
C      LINEAR FINISH EQUAT2 TO START EQUAT3
      XS=XF2
      YS=YF2
      ZS=ZF2
      XF=XS3
      YF=YS3
      ZF=ZS3
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C      START TO FINISH EQUATION 3
```

AD-A190 960 PATH FOLLOWING ROBOT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S G GOODWAY DEC 87 4/4

UNCLASSIFIED F/G 12/9 NL


1·0
2·8
3·15
3·5
4·0
4·5
5·0
2·2
2·0
1·1
1·8
1·25
1·4
1·6

```
      XS=XS3
      YS=YS3
      ZS=ZS3
      XF=XF3
      YF=YF3
      ZF=ZF3
      N=3
      CALL SFPAR(TS,TF,DELT,MAXODX,MAXODY,MAXODZ,J,K,VALMIN,N)
C      LINEAR FINISH EQUAT2 TO START EQUAT3
      XS=XF3
      YS=YF3
      ZS=ZF3
      XF=XS4
      YF=YS4
      ZF=ZS4
      CALL FSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C      START TO FINISH EQUATION 4
      XS=XS4
      YS=YS4
      ZS=ZS4
      XF=XF4
      YF=YF4
      ZF=ZF4
      N=4
      CALL STARF(XS,YS,ZS,XF,YF,ZF,N,
     *   MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C      FINISH EQUAT4 TO HOME
      XF=XF4
      YF=YF4
      ZF=ZF4
      CALL FHOME(XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C      ************************
C      * FINISH POINT FORMATION *
C      ************************
```

E.7 SINUSOIDAL PATH MODIFICATIONS

```
TOF:
C      STEVEN G. GOODWAY
C      ROBPATH7 FORTRAN--PARAMETRIC SUBROUTINE ADDED
C      ************************
C      * USER MODIFICATION AREA *
C      ************************
       FUNCTION  EQUAT1(X,Y)
              EQUAT1=ABS(COS(9*3.141529*X))
              RETURN
       END
C      XY PROJECTION FOR EQUAT1
      FUNCTION EQAT1Y(X)
              EQAT1Y=(ABS(SIN(6*3.141529*X)))
              RETURN
      END
C      END OF FUNCTION DESCRIPTION AREA
C
C      MAIN PROGRAM AREA
C      USER PARAMETER ADJUSTMENTS
C      ***********************
C      * EQUATION BOUNDS AREA  *
C      ***********************
C
C      FIRST PATH EQUATION
C      START/FINISH POINTS
       XS1=1.
       XF1=0.0
C      CALCULATE Y START/FINISH POINTS
       YS1=EQAT1Y(XS1)
       YF1=EQAT1Y(XF1)
C      CALCULATE Z START/FINISH POINTS
       ZS1=EQUAT1(XS1,YS1)
       ZF1=EQUAT1(XF1,YF1)
C      *************************
C      * CALCULATE PATH MOVEMENT *
C      *************************
C
C      PATH FORMATION
       XS=XS1
       YS=YS1
       ZS=ZS1
              XF=XF1
       YF=YF1
       ZF=ZF1
```

```
C     HOME TO START
      CALL HSTAR(XS,YS,ZS,XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C     ***********************
C     * START TO FINISH EQUAT1 *
C     ***********************
     N=1
    CALL STARF(XS,YS,ZS,XF,YF,ZF,N,MAXODX,MAXODY,MAXODZ,VALMIN
   *           ,J,K)
C
C     FINISH EQUAT3 TO HOME
      CALL FHOME(XF,YF,ZF,MAXODX,MAXODY,MAXODZ,VALMIN,J,K)
C
C     ***********************
C     * FINISH POINT FORMATION *
C     ***********************
```

APPENDIX F
3-DIMENSIONAL PLOTTING PROGRAM

This appendix contains the 3-Dimensional plotting program. In the current listing the plane projections will not be drawn. For plane projections just enable all three CURVE functions. A reminder that large data points will require a lot of memory, upwards to 40% of user A-disk, and that parts of the axis may be ignored by the DISSPLA routine. Format of the data points are a integer J, which represents one less then the total of the three tuples in table form, which follow.

```
TOF:
C       STEVEN G. GOODWAY
C       ROBCOMP1.FORTRAN
C       NPS/WEAPONS ENGINEERING/ELECTRICAL ENGINEERING
C       CARTESIAN ROBOT SIMULATION {ROBSIM1.FORTRAN}
C       THIS PROGRAM TAKES THE THREE TUPLE OF POINTS (X,Y,Z)
C       THAT DESCRIBES THE PATH THAT THE ENDPOINT OF A CARTESIAN
C       ROBOT IS TO FOLLOW AND PLOTS THEM IN THREE DIMENSIONS.
C
C
C       MAIN PROGRAM AREA
C       USER PARAMETER ADJUSTMENTS
      DIMENSION POS(1000,3),X(1000),Y(1000),Z(1000)
C
C       *********************
C       * 3-D PLOT SUBROUTINE *
C       *********************
C
      INTEGER N,I
      CALL PAGE(11.,8.5)
```

```
C       GET DATA
      REWIND 02
      READ (2,*) J
        DO 90 I=1,J-1
              READ(2,*)(POS(I,L),L=1,3)
                   X(I)=POS(I,1)
                   Y(I)=POS(I,2)
                   Z(I)=POS(I,3)
 90     CONTINUE
C       OUTPUT TO TEK 618
      CALL TEK618
      CALL NOBRDR
C       SET UP Z-AXIS PARALLEL TO X-Y PLANE
      CALL ZAXANG(-90)
C       SET UP TITLE AND PLOT AREA
      CALL AREA2D(7.,7.)
      CALL HEADIN('3D PATH TRAJECTORY $',100,2.,2)
      CALL HEADIN('ROBOT PATH MOTIONS  $',100,1.5,2)
C       SET UP AXIS LABELS AND DEFINES 3D WORKSPACE
      CALL VOLM3D( 7.,7.,7.)
      CALL X3NAME('X-POS',5)
      CALL Y3NAME('Y-POS',5)
      CALL Z3NAME('Z-POS',5)
C       DEFINE VIEWPOINT
C       CALL VUABS(-30.,-10.,23.)
C       DEFINE THE AXIS SYSTEM
      CALL GRAF3D(0.0,'SCALE',1.0,0.0,'SCALE',1.0,0.0,
     *         'SCALE',1.0)
C       3-D CURVE PLOTTING
      CALL POLY3
      CALL CURV3D(X,Y,Z,J,0)
C       PUT 2-D PLOT ON FLOOR OF 3-D
      CALL GRFITI(0.,0.,0.,1.,0.,0.,0.,1.,0.)
      CALL AREA2D(7.,7.)
      CALL GRAF(0.0,'SCALE',1.0,0.0,'SCALE',1.0)
C       CALL CURVE(X,Y,J,0)
C       PUT 2-D PLOT OF XZ DATA ON BACK WALL OF 3-D
      CALL GRFITI(0.,7.,0.,1.,7.,0.,1.,7.,2.)
      CALL XNAME('X-POS',5)
      CALL AREA2D(7.,7.)
      CALL GRAF(0.0,'SCALE',1.0,0.0,'SCALE',1.0)
C       CALL CURVE(X,Z,J,0)
```

```
C         PUT 2D PLOT OF YZ DATA ON OTHER BACK WALL
      CALL GRFITI(7.,7.,0.,7.,5.,0.,7.,5.,5.)
      CALL XNAME('Y-POS',5)
      CALL YNAME('Z-POS',5)
      CALL AREA2D(7.,7.)
      CALL GRAF(0.0,'SCALE',1.0,0.0,'SCALE',1.0)
C         CALL CURVE(Y,Z,J,0)
C         TERMINATE PLOT
      CALL ENDPL(0)
      CALL DONEPL
      STOP
      END
EOF:
```

APPENDIX G

COORDINATE TRANSFORMATION PROGRAM

This appendix contains the coordinate transformation programs necessary to convert from Cartesian Coordinates to Articulated Coordinates and back to Cartesian Coordinates, with intermediate transformation through spherical coordinate system.

```
 TOF:
C     STEVEN GOODWAY
C     WEAPONS SYSTEMS ENGINEERING/ELECTRICAL ENGINEERING
C     TRANSFORM FORTRAN
C     THE PURPOSE OF THIS PROGRAM IS TO PERFORM COORDINATE
C     TRANSFORMATIONS AND TO PERFORM MISCILANEOUS FUNCTIONS
C     SUCH AS SCALING AND LIMITING FUNCTIONS.  THE COORDINATES
C     TRANSFORMATIONS ARE:
C
C              FROM:          TO:            SUBROUTINE:
C              CARTESIAN      SPHERICAL      CARSP
C              SPHERICAL      CARTESIAN      SPCAR
C              SPHERICAL      ARTICULATED    SPART
C              ARTICULATED    SPHERICAL      ARTSP
C
C     THE MISCELLANEOUS SPECIAL FUNCTIONS ARE:
C              CARTESIAN SCALING             [CARSC]
C              SPHERICAL SCALING             [SPHSC]
C              CARTESIAN LIMITING            [CARLI]
C              SPHERICAL LIMITING            [SPHLI]
C
C   TO OPERATE THE PROGRAM A PATH OF SUBROUTINES MUST BE GENERATED
C     THAT IS IN THE ORDER THAT THE USER INTENDS TO PERFORM IN.
C     ALSO THE USER MUST DECLARE SEVERAL FILEDEFS SO THAT THE FLOW
C     IS PROPER.
C
C     THE FILEDEFS FOR CARTESIAN TO ARTICULATED ARE PRIOR TO
C              SIMULATION:
C                        UNIT    SAMPLE         TYPE
C                        02      PATH4          DATA
C                        03      SCOMP4         DATA
C                        04      COMP4          DATA
```

```
C                    05        SPATH4         DATA
C                    09        TERM
C                    14        APATH4         DATA
C                    15        ASPATH4        DATA
C                    16        SCPATH4        DATA
C
C    THE FILEDEFS FOR ARTICULATED TO CARTESIAN AFTER SIMULATION ARE:
C          UNIT                SAMPLE         TYPE
C          09                  TERM
C          14                  DAPATH4        DATA
C          16                  DCPATH4        DATA
C
C    *************************
C    * MAIN USER PATH FORMATION *
C    *************************
C
C    CARTESIAN TO SPHERICAL
                     REAL MAX
                     MAX=1.0
                     SMAX=0.5
                     SF=1.0
C                    CALL CARLI
C                    CALL CARSC(SF,MAX)
 CC        CALL CARLI
 CC        CALL CARSP
 CCCC SPHERICAL TO ARTICULATED
 CC        SF=1.0
C                    CALL SPHLI
C                    CALL SPHSC(SF,SMAX)
 CC        CALL SPHLI
 CC        CALL SPART
 CCCC      ARTICULATED TO SPHERICAL
                     CALL ARTSP
 CCCC SPHERICAL TO CARTESIAN
 CC        SF=1/SMAX
C                    CALL USPSC(SF)
                     CALL SPCAR
   STOP
   END
C    **************
C    * SUBROUTINES *
C    **************
C
C    *****************
C    * CARSP SUBRUTINE *
C    *****************
```

```
C
C      THIS PROGRAM TRANSFORMS FROM CARTESIAN TO SPHEREICAL
C      COORDINATE SYTEMS.
 C     POS(I,L) = PATH POSITION IN CARTESIAN
 C     SPPOS(I,L) = PATH POSITION IN SPHERICAL
C      HOME FOR THE CARTESIAN ROBOT IS (0.0,0.0,0.0) HOWEVER
C      HOME FOR THE ARTICULATED ROBOT(SPHERICAL) IS (0.5,0.5,0.5).
C      IF INTENDING TO TRANSFORM INSURE THAT THE ORIGINAL PATH
C      STARTS AT THE SPHERICAL HOME POSITION.
               SUBROUTINE CARSP
               INTEGER I,J,K,L
               DIMENSION POS(1000,3),SPPOS(1000,3)
               REWIND 02
C      INPUT CARTESIAN PATH
               READ(2,*) J
               DO 300 I=1,J-1
               READ(2,*)(POS(I,L),L=1,3)
  300          CONTINUE
C      CALCULATIONS FROM CARTESIAN TO SPHERICAL
               DO 400 I=1,J-1
C      CALC RHO
       SPPOS(I,1)=(SQRT((POS(I,1)-0.5)**2+(POS(I,2)-0.5)**2+(
    *                     POS(I,3)-0.5)**2))
C      CALC THETA FROM XY PROJECTION
       B=(SQRT((POS(I,1)-.5)**2+(POS(I,2) -.5)**2))
       A=(SQRT((POS(I,1)-1.)**2+(POS(I,2) -.5)**2))
       C=.5
               PI=3.1415926536
       IF (B.EQ.0.0) THEN
                      SPPOS(I,2)=0.0
       ELSE
                      SPPOS( I,2)=ACOS((B**2+C**2-A**2)/(2.*C*B))
               IF (POS(I,2).LT.0.5) THEN
                      SPPOS(I,2)= 2 * PI -SPPOS(I,2)
               ENDIF
      ENDIF
C      CALC PHI IN THREE DIMENSIONS
               F=(SQRT(B**2+(POS(I,3)-1.)**2))
       R=SPPOS(I,1)
       IF (R.EQ.0.0) THEN
                      SPPOS(I,3)=0.0
       ELSE
                      SPPOS(I,3)=(ACOS((C**2+R**2-F**2)/(2.*C*R)))
       ENDIF
  400          CONTINUE
C      PRINT AND I/O STATEMENTS
```

```
                WRITE(9,*) J,'   CARTESIAN TO SPHERICAL'
                WRITE(5,*) J
        DO 500 I=1,J-1
        WRITE (5,*)(SPPOS(I,L),L=1,3)
  500           CONTINUE
  742           FORMAT (' ',T2,I4,2X,6(F5.3,2X))
       RETURN
       END
C
C       ******************
C       * SPART SUBROUTINE *
C       ******************
C
C       THIS PROGRAM TRANSFORMS FROM SPHERICAL TO ARTICULATED
C       COORDINATE SYTEMS. THE SPHERICAL COORDINATE SYSTEM IS COMPOSED
C       THE THREE TUPLE (RHO,THETA,PHI) AND THE ARTICULATED SYSTEM
C       IS COMPOSED OF THE THREE TUPLE (CP1,CP2,CP3) WHICH DISCRBES
C       THE COMANDED ANGULAR POSITION FOR THE THREE ROBOT MOTIONS AND
C       EXPRESSED IN RADIANS.  TO PROPERLY WORK THE SPHERICAL SYSTEM
C       MUST EXIST WITHIN A 1 UNIT DIAMETER SPHERE.
                SUBROUTINE SPART
                INTEGER I,J,K,L
                DIMENSION ARTPOS(1000,3),SPPOS(1000,3)
C       INPUT CARTESIAN PATH
                REWIND 5
                READ(5,*) J
                DO 301 I=1,J-1
                READ(5,*)(SPPOS(I,L),L=1,3)
  301           CONTINUE
C       CALCULATIONS FROM CARTESIAN TO SPHERICAL
                PI=3.1415926536
                DO 401 I=1,J-1
C       CALCULATE CP1
                ARTPOS(I,1)=SPPOS(I,2)
C       CALCULATE CP2
                ARTPOS(I,2)=PI-SPPOS(I,3)+ACOS(2*SPPOS(I,1))
C       CALCULATE CP3
                ARTPOS(I,3)=PI-2*(ACOS(2*SPPOS(I,1)))
  401           CONTINUE
C       PRINT AND I/O
                REWIND 14
                WRITE(9,*) 'SPHERICAL TO ARTICULATED (R,T,P,CP1,CP2,CP3)'
                WRITE(14,*) J
                DO 501 I=1,J-1
                WRITE (14,*)(ARTPOS(I,L),L=1,3)
  501           CONTINUE
```

```
 742          FORMAT (' ',T2,I4,2X,6(F5.3,2X))
   RETURN
   END
C
 C     *******************
 C     * ARTSP SUBROUTINE *
 C     *******************
C
C      THIS PROGRAM TRANSFORMS FROM ARTICULATED TO SPHERICAL
C      COORDINATE SYTEMS. THE SPHERICAL COORDINATE SYSTEM IS COMPOSED
C      THE THREE TUPLE (RHO,THETA,PHI) AND THE ARTICULATED SYSTEM
C      IS COMPOSED OF THE THREE TUPLE (CP1,CP2,CP3) WHICH DESCRIBES
C      THE COMANDED ANGULAR POSITION FOR THE THREE ROBOT MOTIONS AND
C      EXPRESSED IN RADIANS.  TO PROPERLY WORK THE SPHERICAL SYSTEM
C      MUST EXIST WITHIN A 1 UNIT DIAMETER SPHERE.
              SUBROUTINE ARTSP
              INTEGER I,J,K,L
              DIMENSION ARTPOS(1000,3),SPPOS(1000,3)
C      INPUT CARTESIAN PATH
              REWIND 14
              READ(14,*) J
              DO 302 I=1,J-1
              READ(14,*)(ARTPOS(I,L),L=1,3)
 302          CONTINUE
C      CALCULATIONS FROM ARTICULATED TO SPHERICAL
              PI=3.1415926536
              DO 402 I=1,J-1
C      CALCULATE RHO
              SPPOS(I,1)=.5*COS((PI-ARTPOS(I,3))/2)
C      CALCULATE THETA
              SPPOS(I,2)=ARTPOS(I,1)
C      CALCULATE PHI
              SPPOS(I,3)=PI-ARTPOS(I,2)+(ACOS(2*SPPOS(I,1)))
 402          CONTINUE
C      PRINT AND I/O
              REWIND 15
              WRITE(9,*) 'ARTICULATED TO SPHERICAL (CP1,CP2,CP3,R,T,P)'
              WRITE(15,*) J
              DO 502 I=1,J-1
              WRITE (15,*)(SPPOS(I,L),L=1,3)
 502          CONTINUE
 742          FORMAT (' ',T2,I4,2X,6(F5.3,2X))
   RETURN
   END
C
 C     ******************
```

```
C     * SPCAR SUBROUTINE *
C     ******************
C
C     THIS PROGRAM TRANSFORMS FROM SPHERICAL TO CARTESIAN
C     COORDINATE SYTEMS. THE SPHERICAL COORDINATE SYSTEM IS COMPOSED
C     THE THREE TUPLE (RHO,THETA,PHI) AND THE CARTESIAN SYSTEM
C     IS COMPOSED OF THE THREE TUPLE (X,Y,Z) WHICH DESCRIBES
C     THE COORDINATES THAT WERE SIMULATED BY THE ROBOT MOTION.
              SUBROUTINE SPCAR
              INTEGER I,J,K,L
              DIMENSION POS(1000,3),SPPOS(1000,3)
C     INPUT CARTESIAN PATH
              REWIND 15
              READ(15,*) J
              DO 303 I=1,J-1
              READ(15,*)(SPPOS(I,L),L=1,3)
 303          CONTINUE
C     CALCULATIONS FROM SPHERICAL TO CARTESIAN
              DO 403 I=1,J-1
C     CALCULATE X
              R=SPPOS(I,1)*SIN(SPPOS(I,3))
              POS(I,1)=.5+R*COS(SPPOS(I,2))
C     CALCULATE Y
              POS(I,2)=.5+R*SIN(SPPOS(I,2))
C             CALCULATE Z
              POS(I,3)=.5+SPPOS(I,1)*COS(SPPOS(I,3))
 403          CONTINUE
C     PRINT AND I/O
              REWIND 16
              WRITE(9,*) 'SPHERICAL TO CARTESIAN (R,T,P,X,Y,Z)'
              WRITE(16,*) J
              DO 503 I=1,J-1
              WRITE (16,*)(POS(I,L),L=1,3)
 503          CONTINUE
 742          FORMAT (' ',T2,I4,2X,6(F5.3,2X))
      RETURN
      END
C
C     ******************************
C     * SCALE IN CARTESIAN COORDNATES *
C     ******************************
C
C     THE PURPOSE OF THIS PROGRAM IS TO TAKE A SET OF CARTESIAN
C     COORDINATE THREE TUPLE AND SCALE SO THAT THE LARGEST VALUE
C     IS ON THE SURFACE OF A (1,1,1) CUBE. THIS INSURES THAT THE
C     MAXIMUM ATTEMPTED MOVEMENT IS WITHIN THE RANGE OF THE MOTION
```

```
C       OF THE ROBOT ARM WHEN THIS "NORMALIZED" VALUE IS USED WITH
C       A SCALING FACTOR (SF).  WITH SF LESS THAN OR EQUAL TO ONE THEN
C       THE MOTION WILL BE WITHIN THE UNIT CUBE.  A SF GREATER THAN
C       ONE CAN BE USED WITH A CARTESIAN ROBOT ARM WHICH REQUIRES
C       INPUTS TO BE SCALED TO THE ARM LENGTHS.  WHEN USED IN
C       CONJUNCTION WITH THE SPHERICAL PROGRAM THE CARTESIAN COORD
C       MUST BE WITHIN THE UNIT CUBE.
              SUBROUTINE CARSC(SF,MAX)
              INTEGER I,J,K,L
              REAL SF,MAX
              DIMENSION POS(1000,3)
              REWIND 2
C       INPUT CARTESIAN PATH
              READ(2,*) J
              DO 305 I=1,J-1
              READ(2,*)(POS(I,L),L=1,3)
              IF (POS(I,1).GT.MAX) THEN
                              MAX=POS(I,1)
              ENDIF
              IF (POS(I,2).GT.MAX) THEN
                              MAX=POS(I,2)
              ENDIF
              IF (POS(I,3).GT.MAX) THEN
                              MAX=POS(I,3)
              ENDIF
  305         CONTINUE
C       SCALE
              IF (MAX.EQ.0.0) THEN
              GOTO 455
              ENDIF
              DO 405 I=1,J-1
              POS(I,1)=POS(I,1)*SF/MAX
              POS(I,2)=POS(I,2)*SF/MAX
              POS(I,3)=POS(I,3)*SF/MAX
  405         CONTINUE
C       PRINT AND I/O
              REWIND 02
              WRITE (9,*) 'CARTESIAN SCALING  SF=',SF
              WRITE (2,*) J
              DO 505 I=1,J-1
              WRITE(2,*)(POS(I,L),L=1,3)
              WRITE(9,743) I,(POS(I,L),L=1,3)
  505         CONTINUE
  743         FORMAT(' ',T2,I4,2X,3(F5.3,2X))
  455         RETURN
    END
```

```
C
C       ******************************
C       * SCALE IN SPHERICAL COORDNATES *
C       ******************************
C
C       THE PURPOSE OF THIS PROGRAM IS TO TAKE A SET OF SPHERICAL
C       COORDINATE THREE TUPLE AND SCALE RHO SO THAT THE LARGEST VALUE
C       IS ON THE SURFACE OF A UNIT SPHERE.  THIS INSURES THAT THE
C       MAXIMUM ATTEMPTED MOVEMENT IS WITHIN THE RANGE OF THE MOTION
C       OF THE ROBOT ARM WHEN THIS "NORMALIZED" VALUE IS USED WITH
C       A SCALING FACTOR (SF).  WITH SF LESS THAN OR EQUAL TO ONE THEN
C       THE MOTION WILL BE WITHIN THE UNIT SPHERE.  A SF GREATER THAN
C       ONE CAN BE USED WITH THE ARTICULATED ROBOT ARM WHICH REQUIRES
C       INPUTS TO BE SCALED TO THE ARM LENGTHS.  WHEN USED IN
C       CONJUNCTION WITH THE ARTICULATED PROGRAM THE SHPERICAL COORD
C       MUST BE WITHIN THE UNIT SPHERE.  A UNIT SPHERE IS DEFINED TO
C       HAVE A DIAMETER OF 1 UNIT.
                SUBROUTINE SPHSC(SF,MAX)
                INTEGER I,J
                REAL SF,MAX
                DIMENSION SPPOS(1000,3)
                REWIND 5
C       INPUT SPHERICAL PATH
                READ(5,*) J
                DO 306 I=1,J-1
                READ(5,*)(SPPOS(I,L),L=1,3)
                IF (SPPOS(I,1).GT.MAX) THEN
                                MAX=SPPOS(I,1)
                ENDIF
  306           CONTINUE
C       SCALE
                IF (MAX.EQ.0.0) THEN
                GOTO 456
                ENDIF
                 DO 406 I=1,J-1
                SPPOS(I,1)=SPPOS(I,1)*SF/(2*MAX)
  406           CONTINUE
C       PRINT AND I/O
                REWIND 05
                 WRITE (9,*) 'SPHERICAL SCALING  SF=',SF,'  MAX =', MAX
                WRITE (5,*) J
                DO 506 I=1,J-1
                WRITE(5,*)(SPPOS(I,L),L=1,3)
  506           CONTINUE
  743           FORMAT(' ',T2,I4,2X,3(F5.3,2X))
  456           RETURN
```

```
   END
C
C     *******************************
C     * UNSCALE IN SPHERICAL COORDNATES *
C     *******************************
C
C     THE PURPOSE OF THIS PROGRAM IS TO TAKE A SET OF SPHERICAL
C     COORDINATE THREE TUPLE AND UNSCALE RHO SO THAT THE VALUES ARE
C     CLOSE TO WHAT THEY WERE PRIOR TO SENDING THEM TO THE ROBOT
C     ARM.  THIS WAY THE FINAL RESULT IN CARTESIAN WILL COMPARE WITH
C     THE VALUES THAT THEY STARTED WITH. THIS IS DONE BY USING
C     A SCALING FACTOR (SF).
              SUBROUTINE USPSC(SF)
              INTEGER I,J
              REAL SF
              DIMENSION SPPOS(1000,3)
              REWIND 15
C     INPUT SPHERICAL PATH
              READ(15,*) J
              DO 306 I=1,J-1
              READ(15,*)(SPPOS(I,L),L=1,3)
  306         CONTINUE
C     SCALE
              DO 406 I=1,J-1
              SPPOS(I,1)=SPPOS(I,1)*SF
  406         CONTINUE
C     PRINT AND I/O
              REWIND 15
              WRITE (9,*) 'SPHERICAL UN-SCALING  SF=',SF
              WRITE (15,*) J
              DO 506 I=1,J-1
              WRITE(15,*)(SPPOS(I,L),L=1,3)
  506         CONTINUE
  743         FORMAT(' ',T2,I4,2X,3(F5.3,2X))
  456         RETURN
   END
C
C     *****************************
C     * LIMIT IN CARTESIAN COORDNATES *
C     *****************************
C
C     THE PURPOSE OF THIS PROGRAM IS TO TAKE A SET OF CARTESIAN
C     COORDINATE THREE TUPLE AND LIMIT VALUES LARGER THAN ONE SO THEY
C     ARE ON THE SURFACE OF A (1,1,1) CUBE.  THIS INSURES THAT THE
C     MAXIMUM ATTEMPTED MOVEMENT IS WITHIN THE RANGE OF THE MOTION
C     OF THE ROBOT ARM.
```

```
                  SUBROUTINE CARLI
                  INTEGER I,J
                  DIMENSION POS(1000,3)
                  REWIND 2
C       INPUT CARTESIAN PATH
                  READ(2,*) J
                  DO 307 I=1,J-1
                  READ(2,*)(POS(I,L),L=1,3)
        IF (POS(I,1).GT.1.0) THEN
                           POS(I,1)=1.0
        ENDIF
        IF (POS(I,2).GT.1.0) THEN
                           POS(I,2)=1.0
        ENDIF
        IF (POS(I,3).GT.1.0) THEN
                           POS(I,3)=1.0
        ENDIF
  307             CONTINUE
C       PRINT AND I/O
                  REWIND 02
                  WRITE (9,*) 'CARTESIAN LIMITING'
                  WRITE (2,*) J
                  DO 507 I=1,J-1
                  WRITE(2,*)(POS(I,L),L=1,3)
  507             CONTINUE
  743             FORMAT(' ',T2,I4,2X,3(F5.3,2X))
      RETURN
      END
C
C       ******************************
C       * LIMIT IN SPHERICAL COORDNATES *
C       ******************************
C
C       THE PURPOSE OF THIS PROGRAM IS TO TAKE A SET OF SPHERICAL
C       COORDINATE THREE TUPLE AND LIMIT RHO SO THAT THE LARGEST VALUE
C       IS ON THE SURFACE OF A UNIT SPHERE.  THIS INSURES THAT THE
C       MAXIMUM ATTEMPTED MOVEMENT IS WITHIN THE RANGE OF THE MOTION
C       OF THE ROBOT ARM.
                  SUBROUTINE SPHLI
                  INTEGER I,J
                  DIMENSION SPPOS(1000,3)
                  REWIND 5
C       INPUT SPHERICAL PATH
                  READ(5,*) J
                  DO 308 I=1,J-1
                  READ(5,*)(SPPOS(I,L),L=1,3)
        IF (SPPOS(I,1).GT.0.5) THEN
                      SPPOS(I,1)=0.5
        ENDIF
  308          CONTINUE
C       PRINT AND I/O
               REWIND 05
               WRITE (9,*) 'SPHERICAL LIMITING'
               WRITE (5,*) J
               DO 508 I=1,J-1
               WRITE(5,*)(SPPOS(I,L),L=1,3)
 508           CONTINUE
  743          FORMAT(' ',T2,I4,2X,3(F5.3,2X))
     RETURN
     END
 EOF:
```

APPENDIX H

MISCILLANEOUS PLOTTINGS

This appendix contains miscillaneous plots that were not included in the text but may be of interest to the reader.

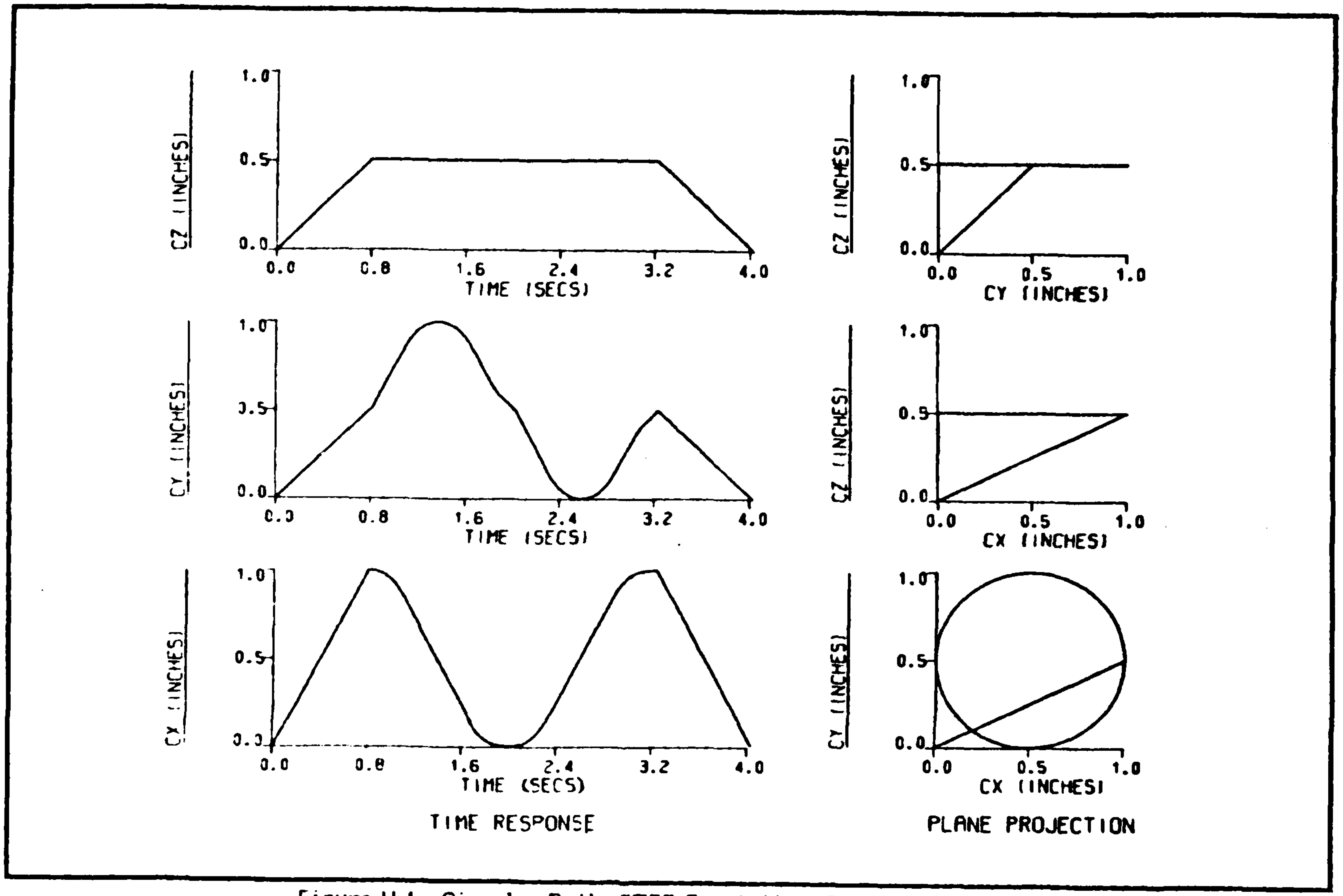


Figure H.1 Circular Path, STEP Excitation, Position Method
(a) X, Y, and Z Time Response, (b) XY, XZ, and YZ Plane Projections.

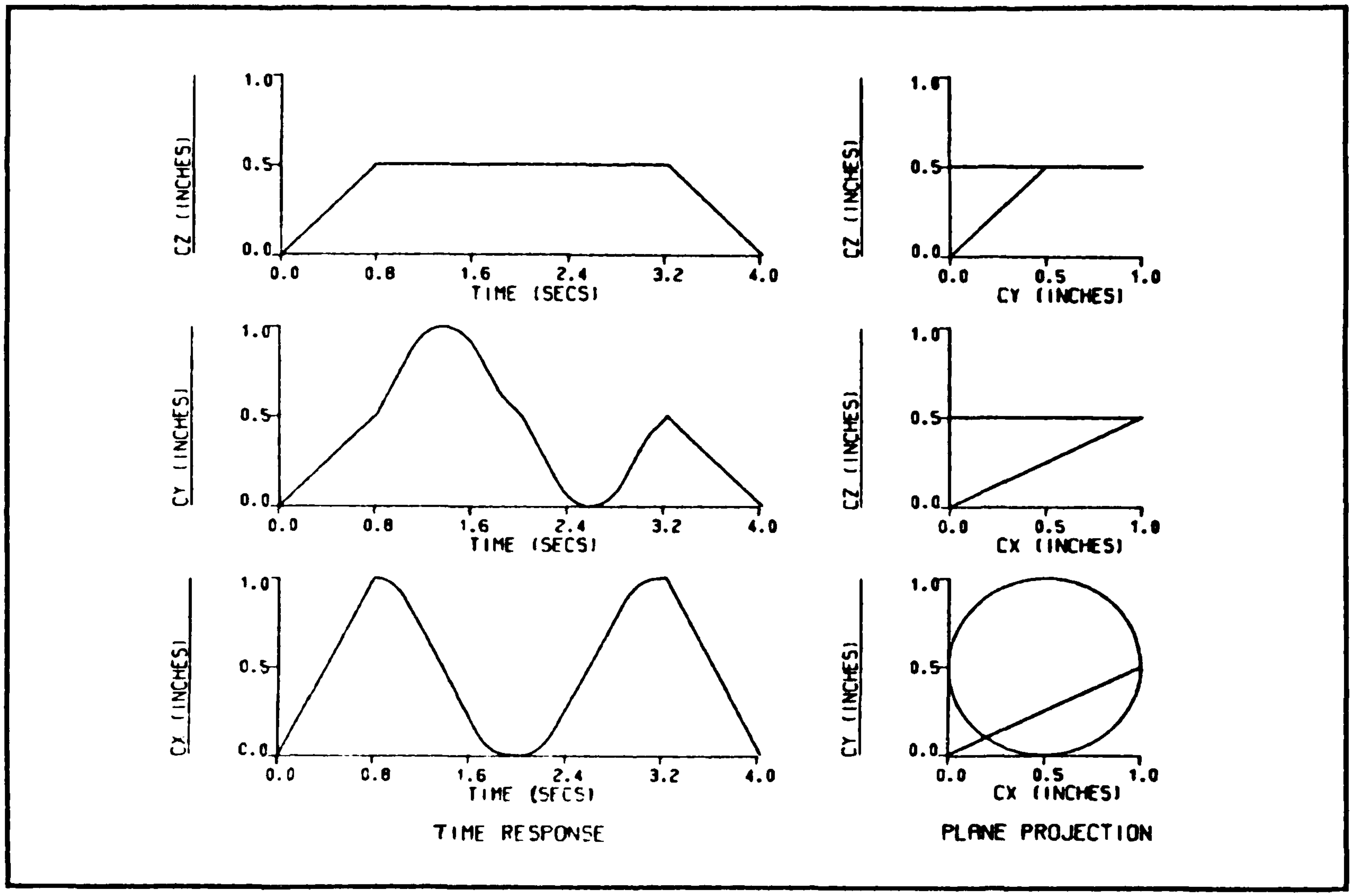


Figure H.2 Circular Path, RAMP Excitation, Position Method
(a) X, Y, and Z Time Response, (b) XY, XZ, and YZ Plane Projections.

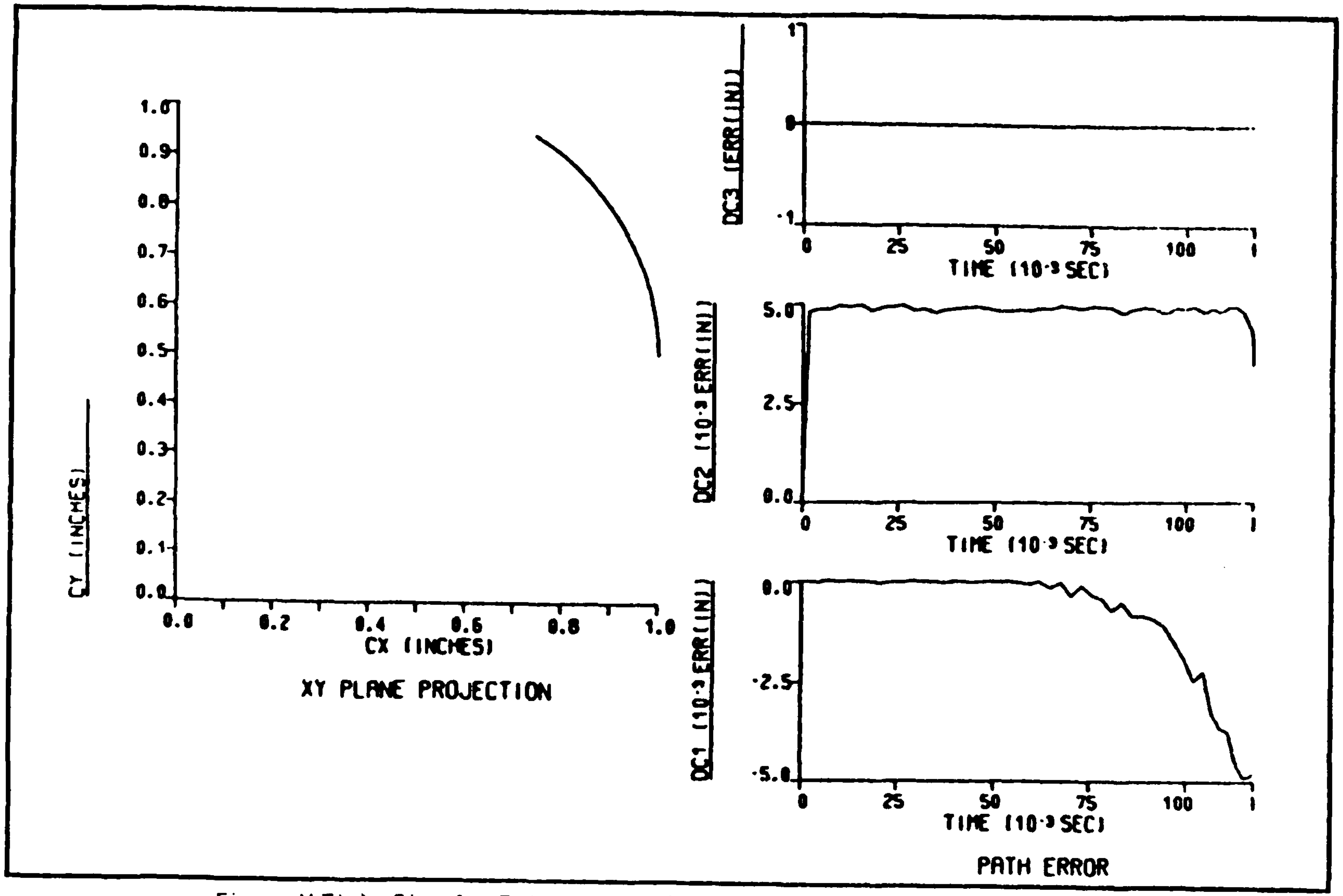


Figure H.3(a) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 10.0 milli-in. XY Plane Projection and X,Y, and Z Path Errors.

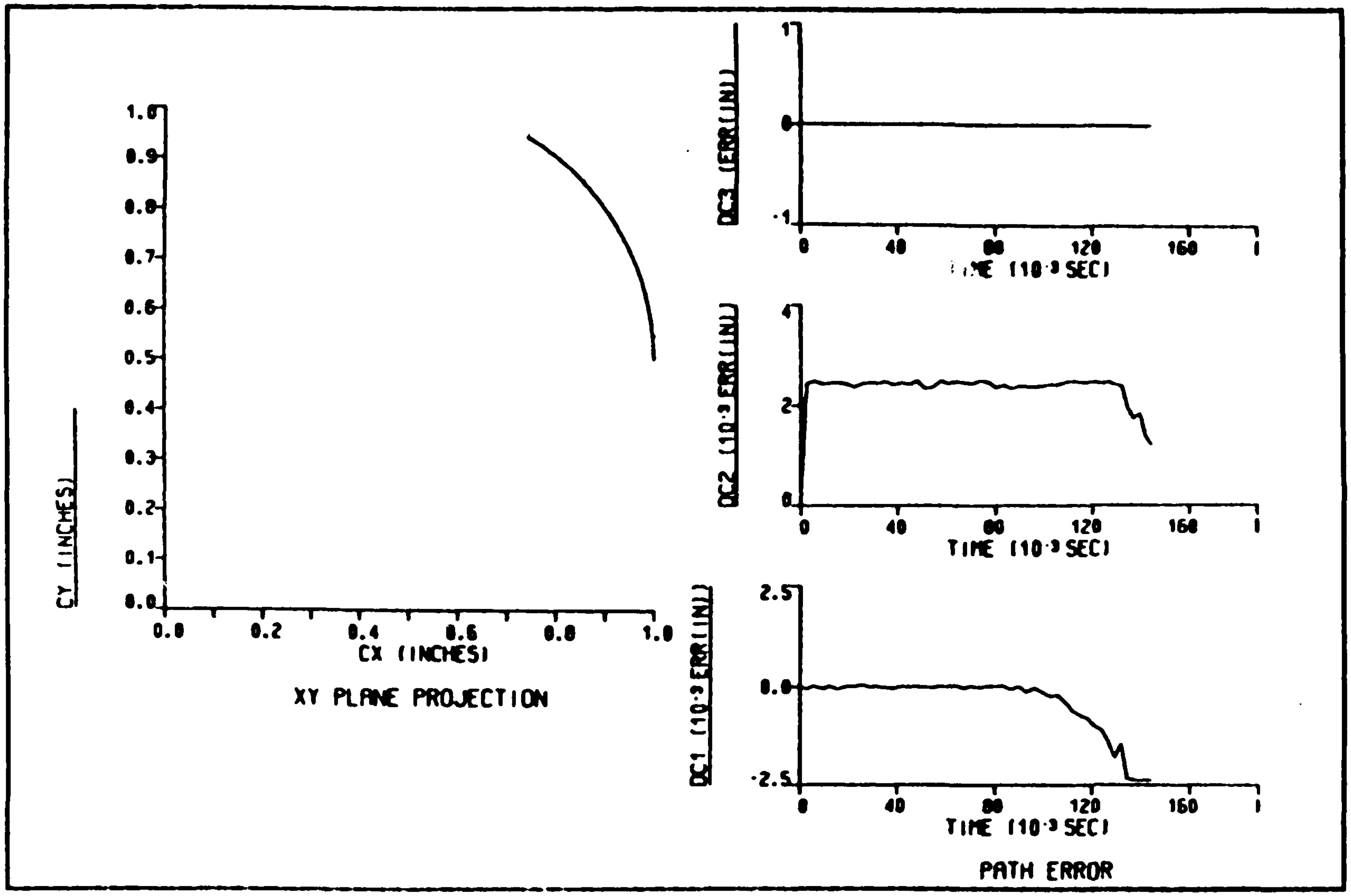


Figure H.3(b) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 5.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

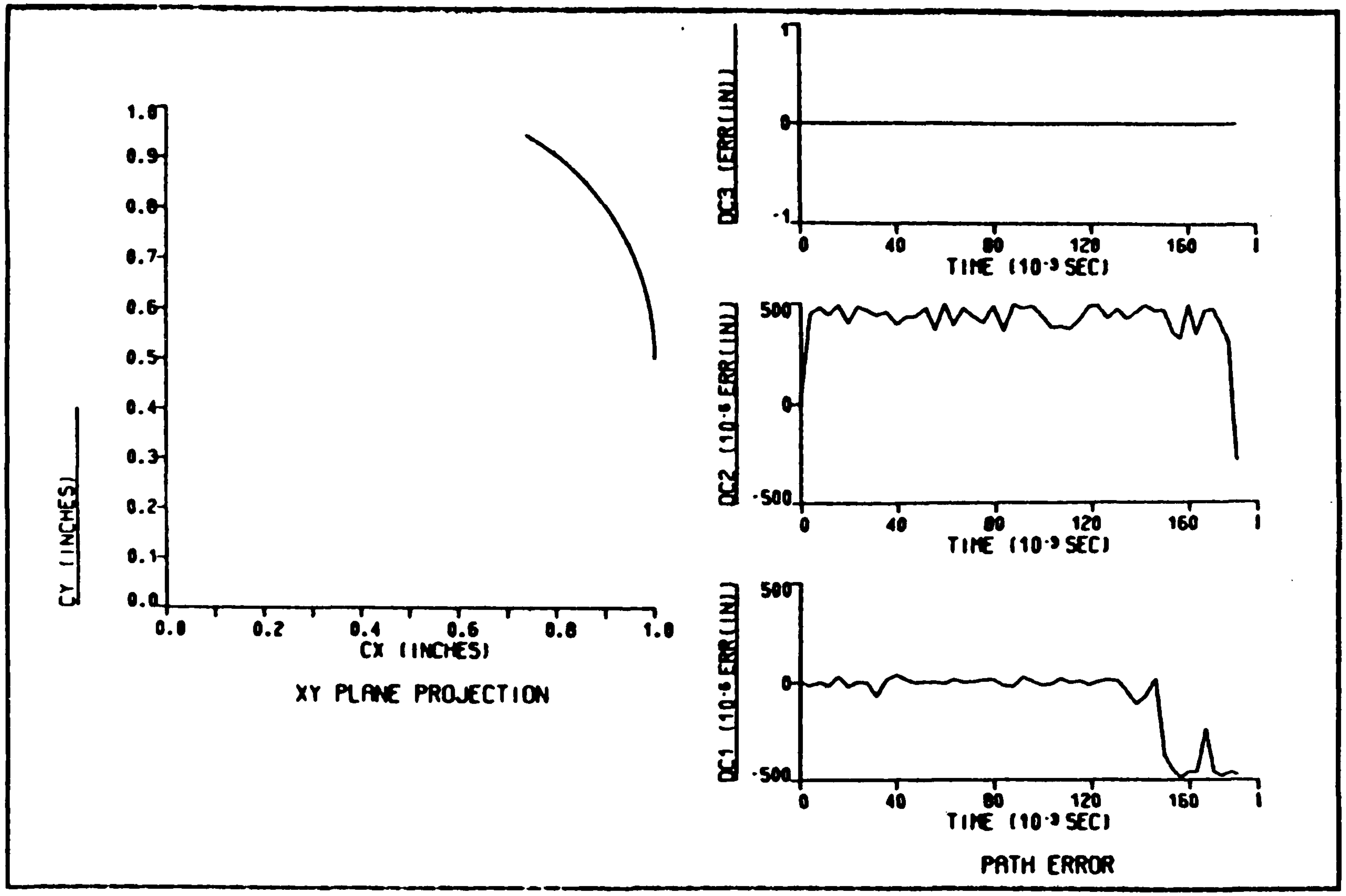


Figure H.3(c) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 1.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

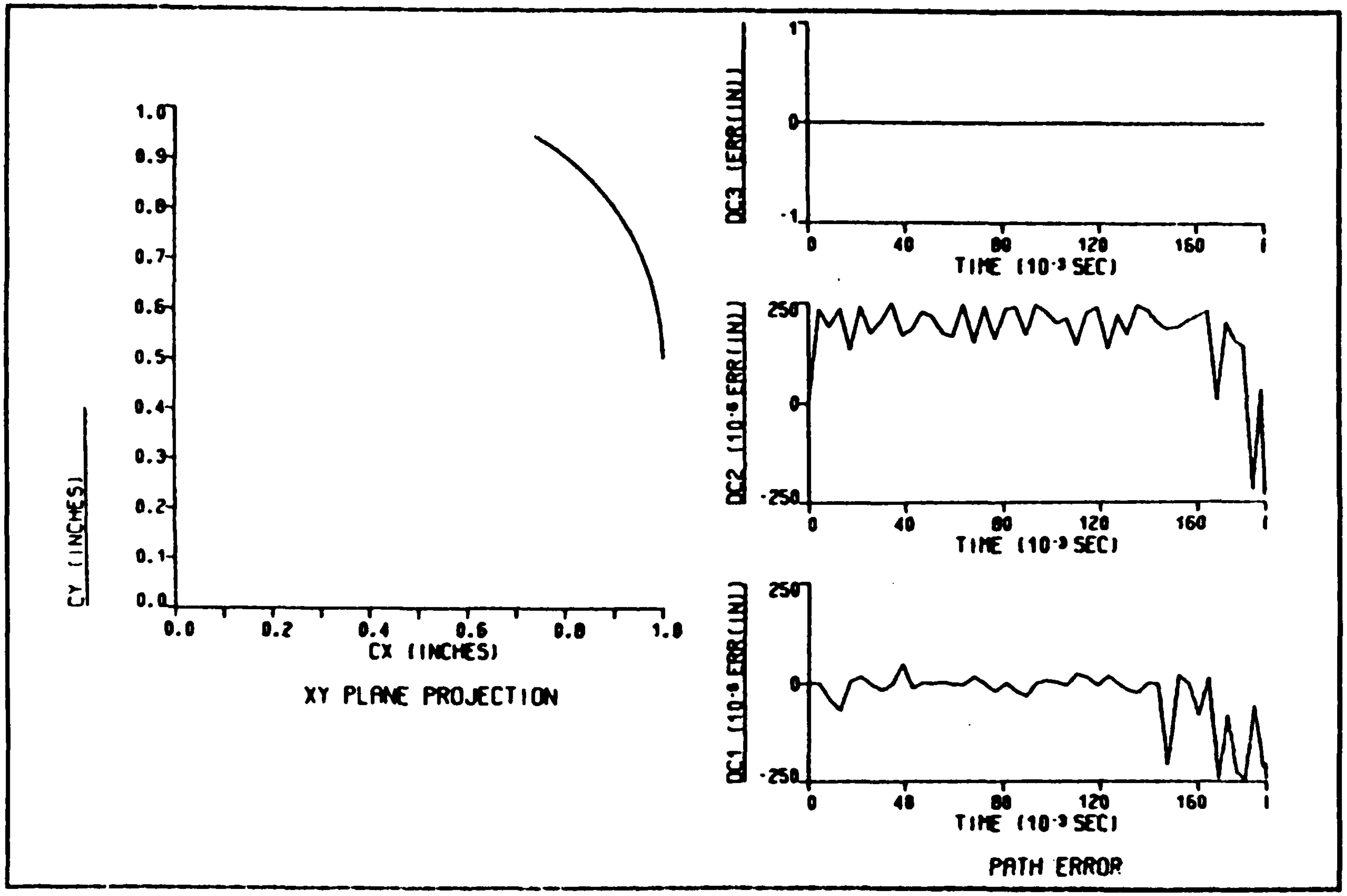


Figure H.3(d) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 0.5 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

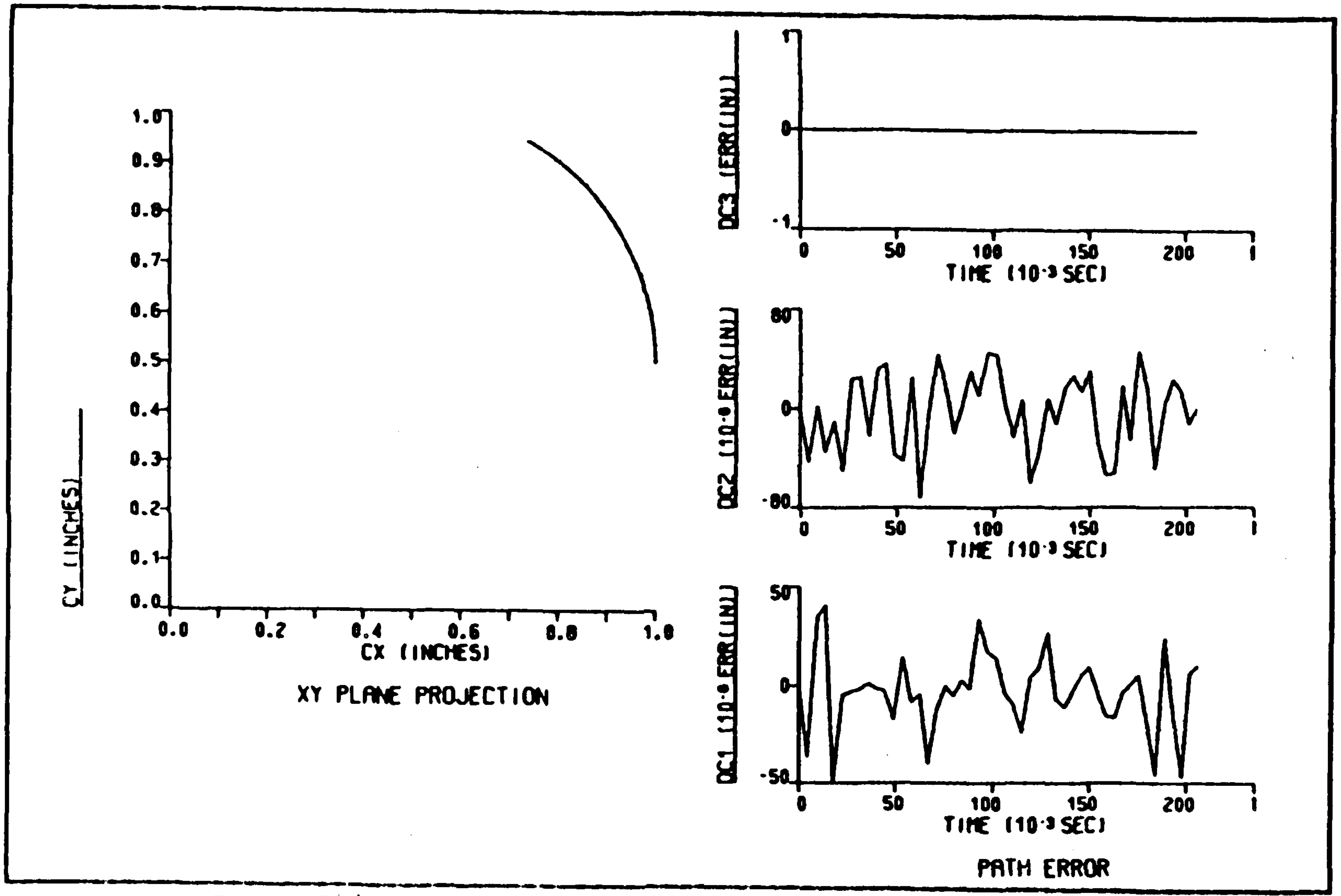


Figure H.3(e) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 0.1 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

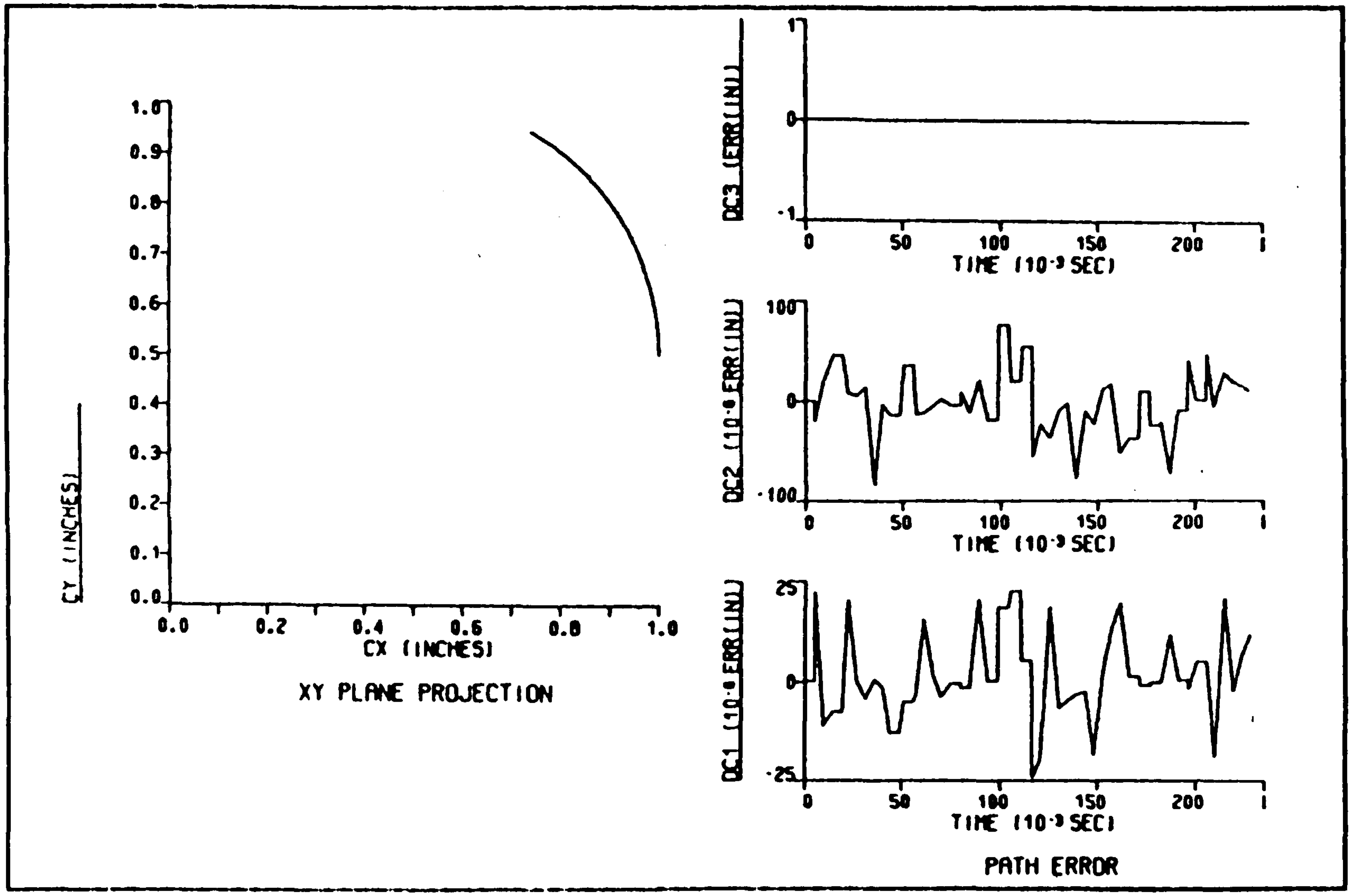


Figure H.3(f) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 0.05 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

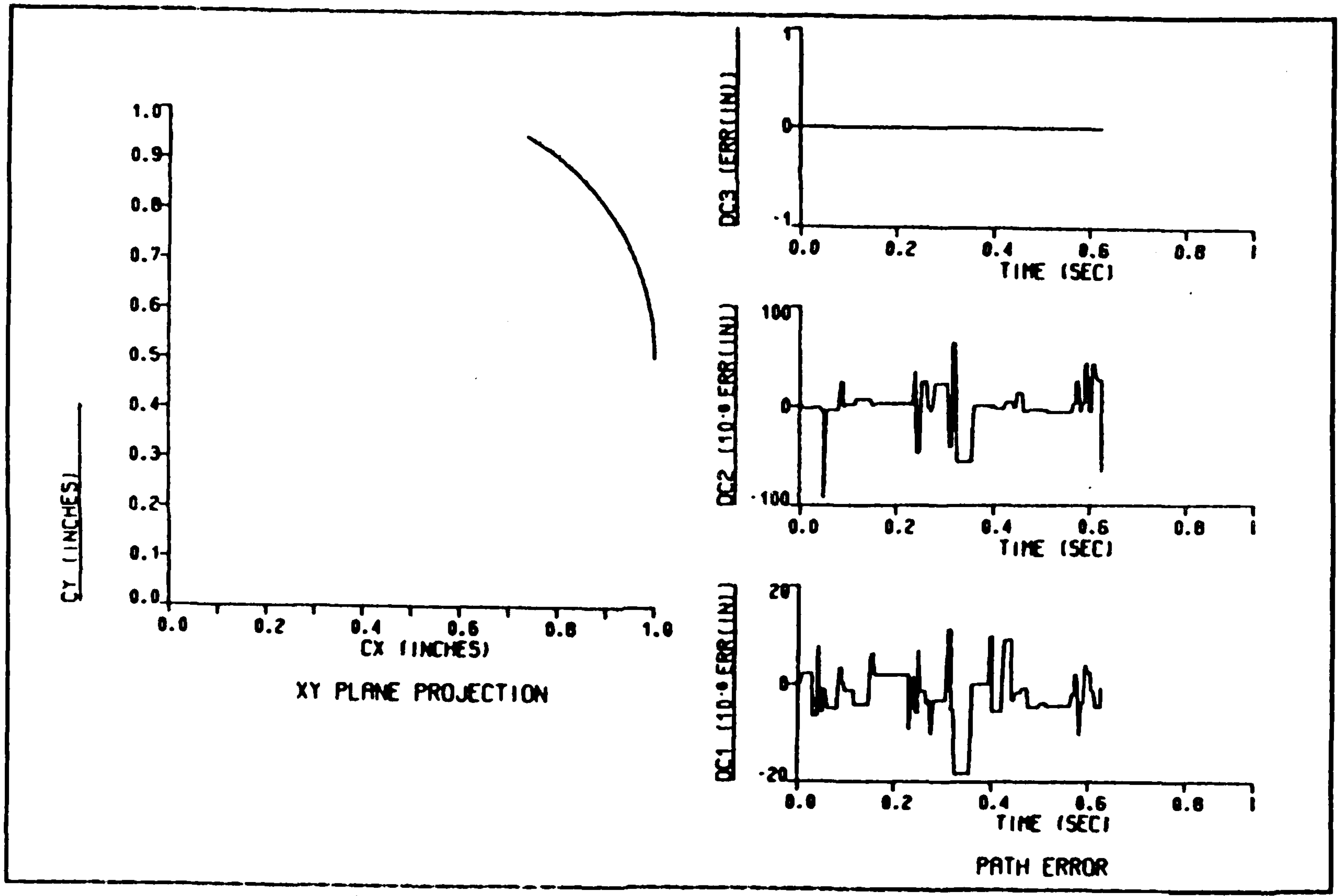


Figure H.3(g) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 0.01 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

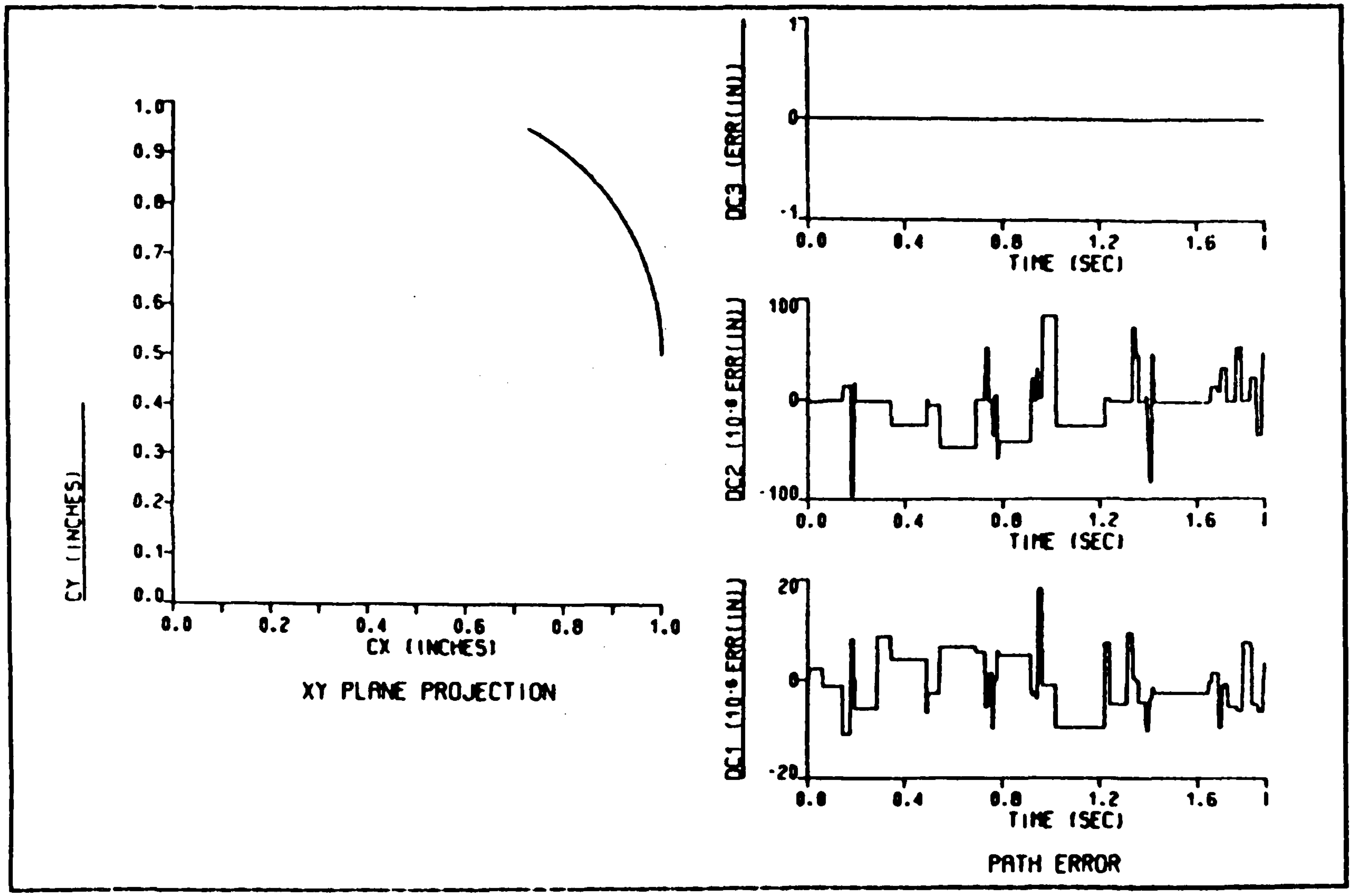


Figure H.3(h) Circular Path (Reduced), STEP Excitation, Velocity Method, Valmin = 0.005 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

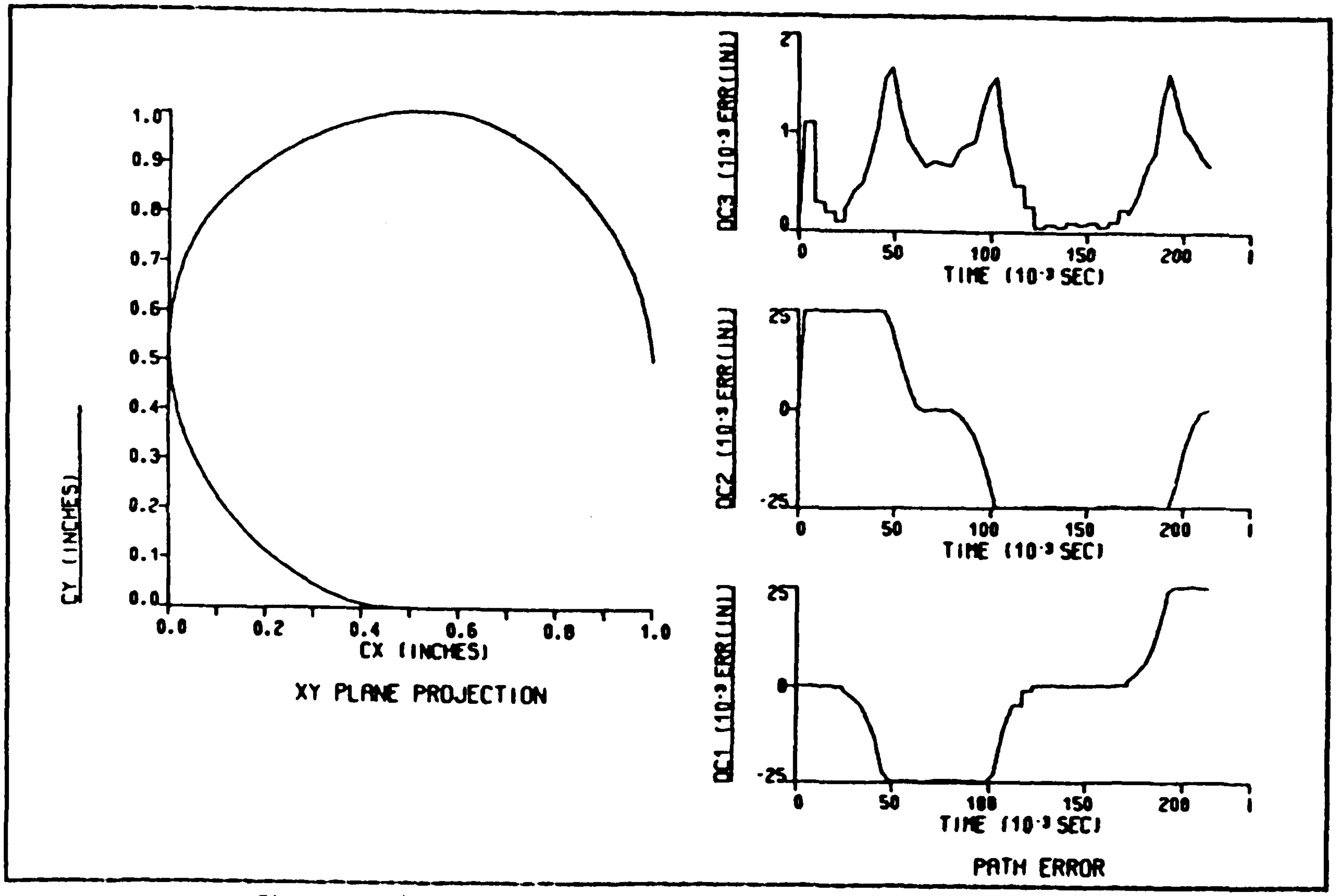


Figure H.4(a) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 50.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors

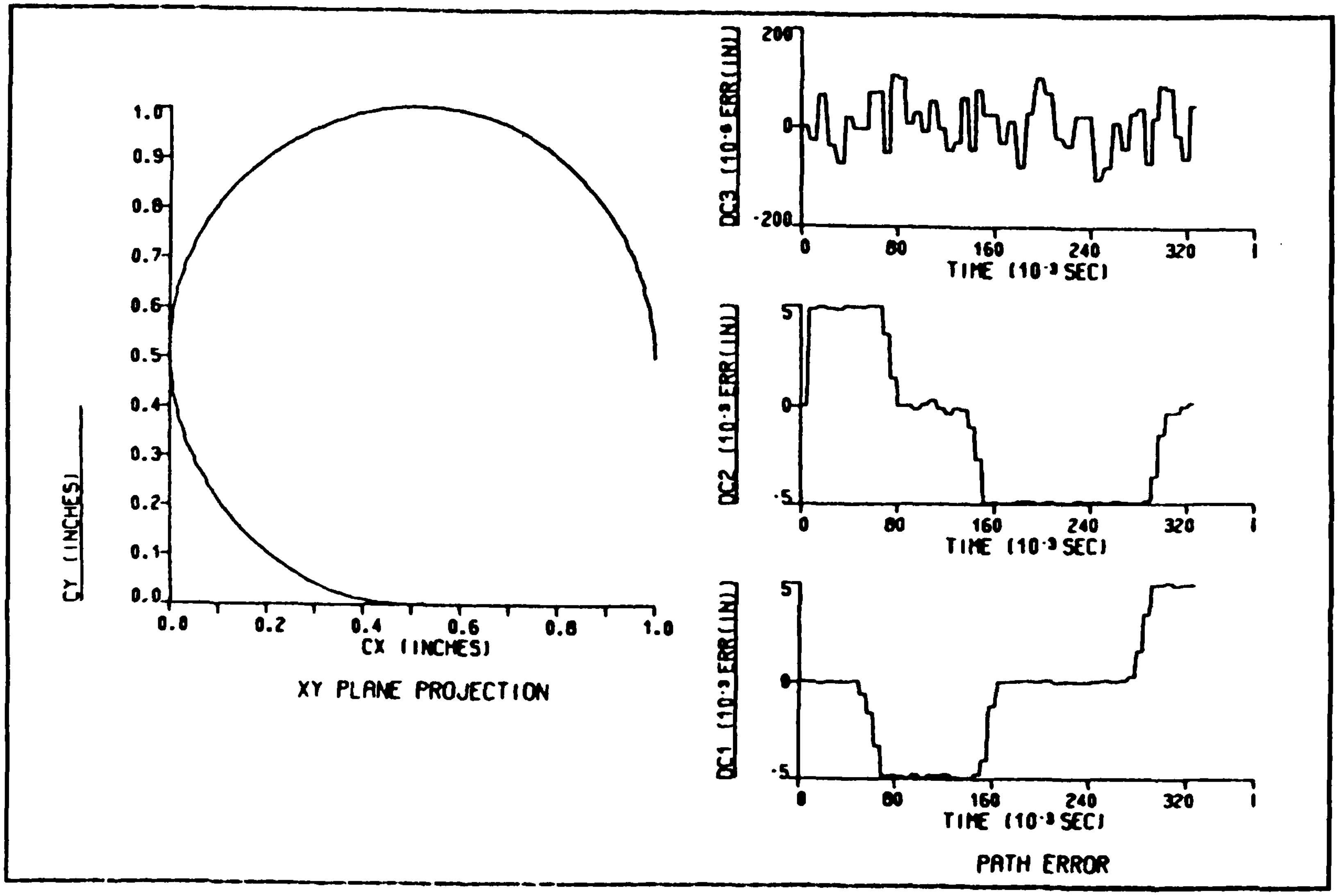


Figure H.4(b) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Volmin = 10.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

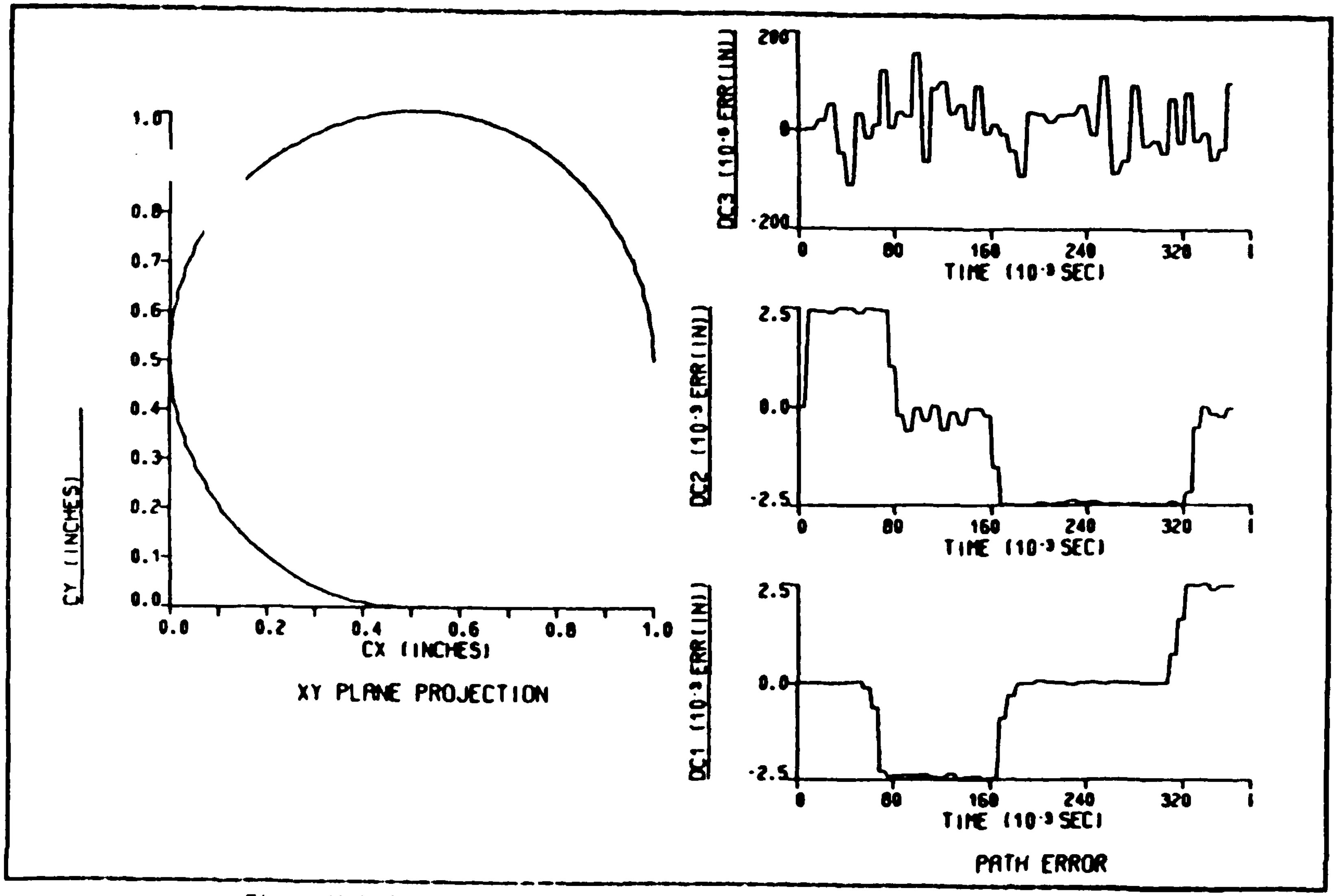


Figure H.4(c) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 5.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

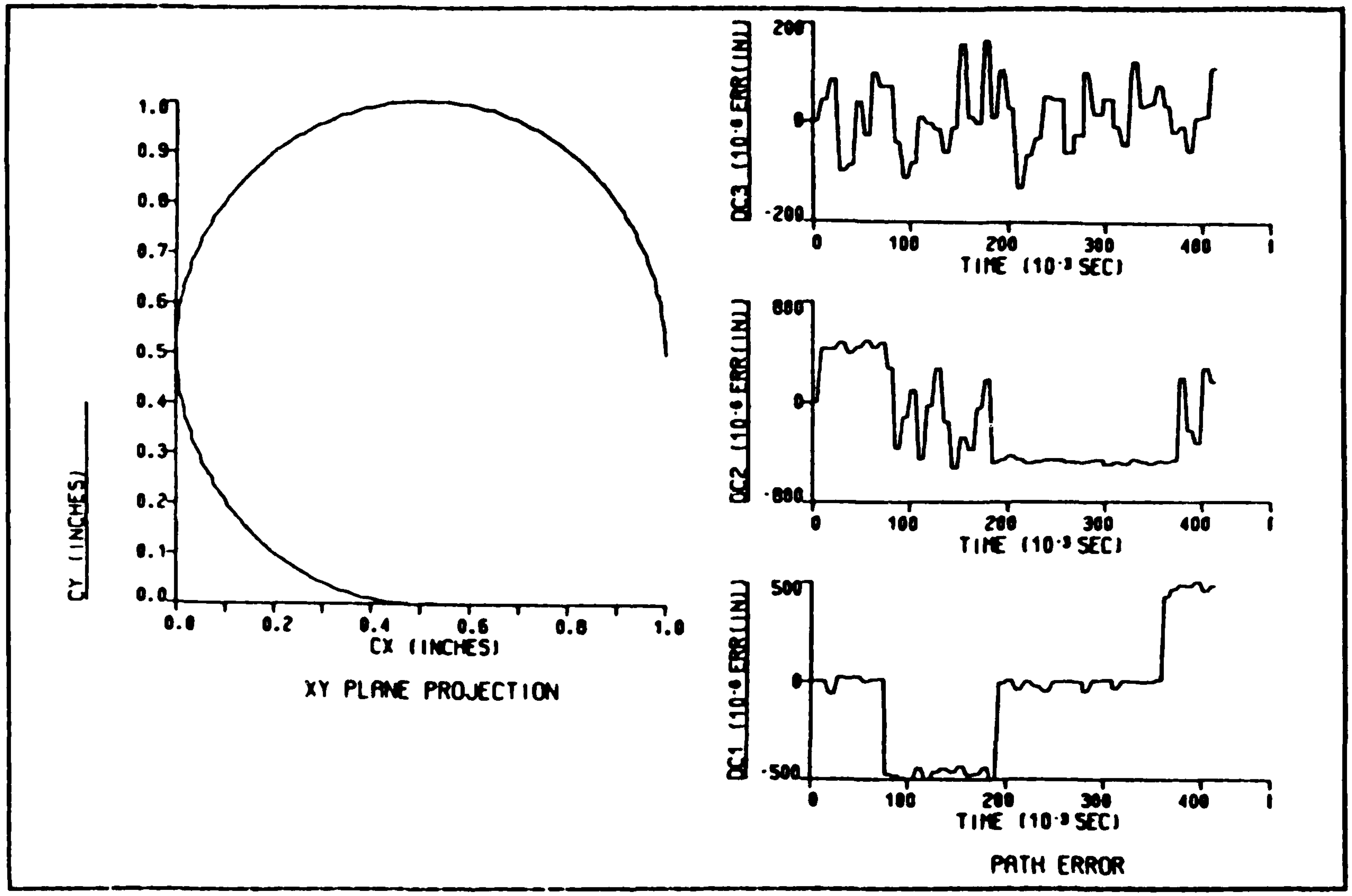


Figure H 4(d) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 1.0 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

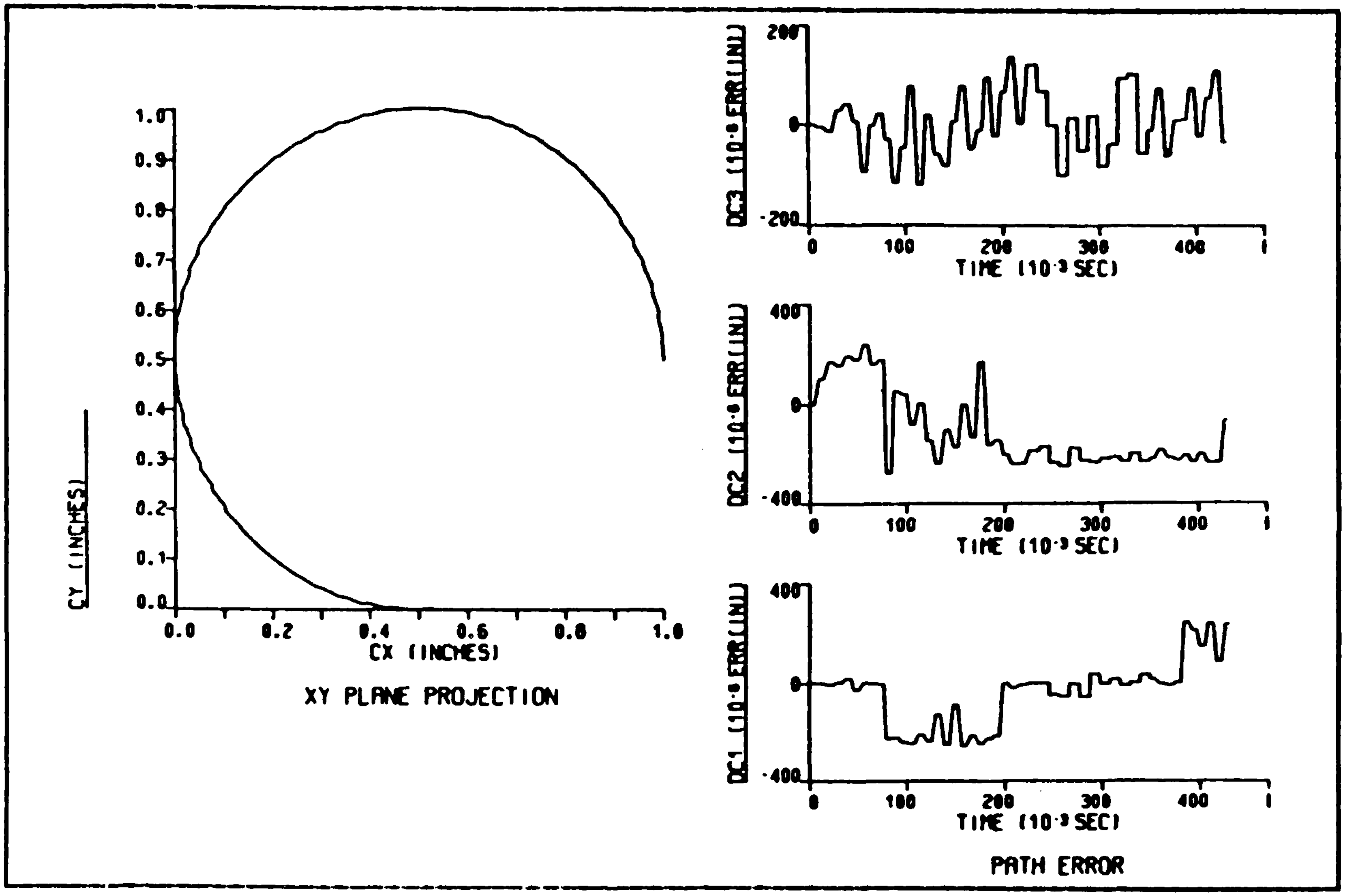


Figure H 4(e) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 0.5 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

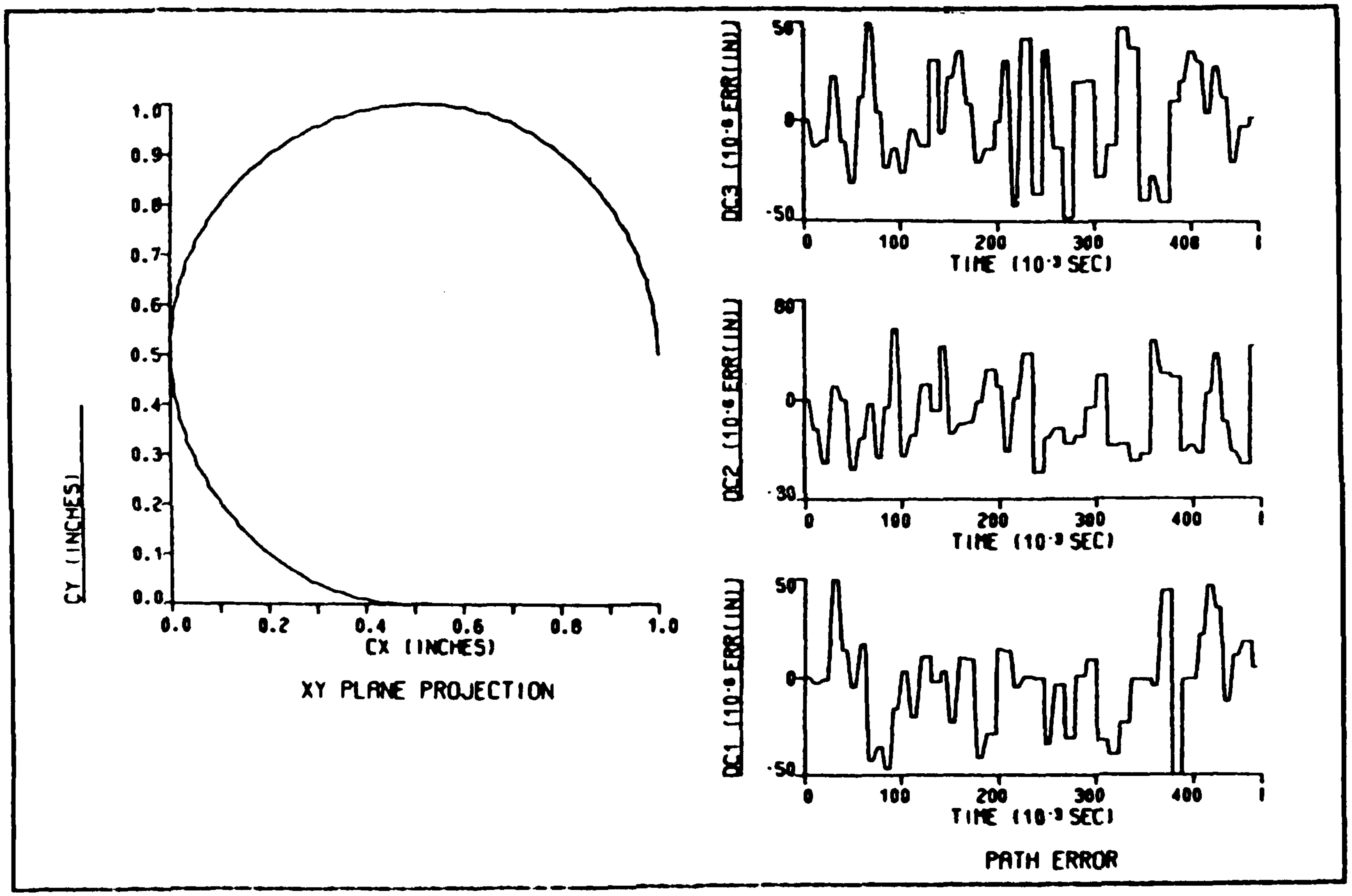


Figure H.4(f) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 0.1 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

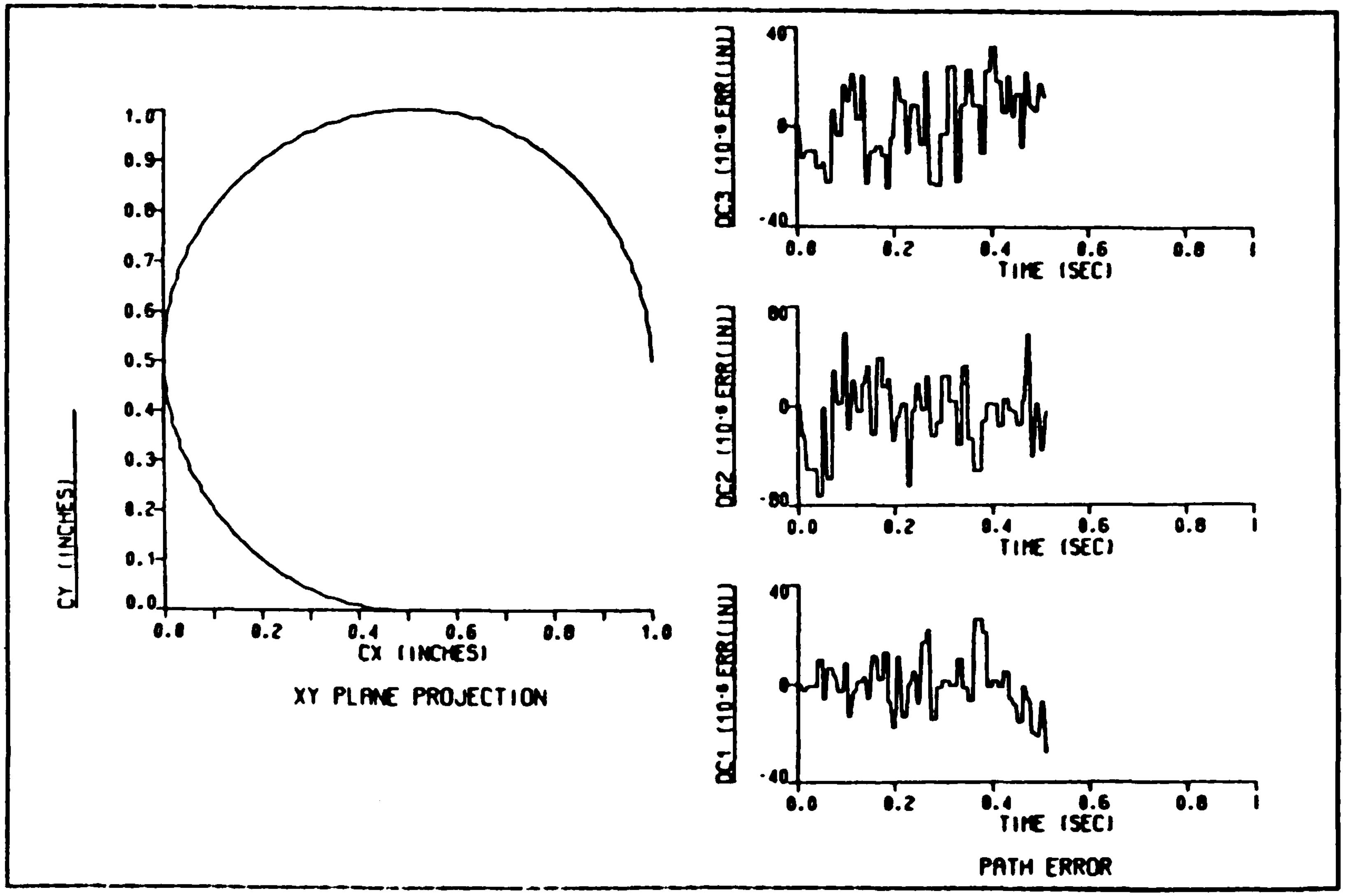


Figure H 4(g) Helix Path 1 (Reduced), STEP Excitation, Velocity Method, Valmin = 0.05 milli-in, XY Plane Projection and X,Y, and Z Path Errors.

LIST OF REFERENCES

Brady, M., and others, Robot Motion: Planning and Control, MIT Press, 1982.

Coiffet,P., Robot Technology: Modeling and Control, Prentice-Hall, Inc, 1983.

Featherstone, R., "Position and Velocity Transformations Between Robot End-Effector Coordinates and Joint Angles," International Journal of Robotics Research, v. 2, no. 2, Summer 1983.

Fu, K.S., Gonzalez, R.C., and Lee, C.S.G., Robotics:Control, Sensing, Vision, and Intelligence, McGraw-Hill Book Co., 1987.

Kalogiros, G., Automatic Control of Robot Motion , Masters Thesis, Naval Postgraduate School, Monterey, CA, December 1987.

Koran, Y., Robotics for Engineers, McGraw-Hill Book Co., 1985.

Lee, C.S.G., "Robot Arm Kinematics, Dynamics, and Control," Computer, December 1982.

Lee, C.S.G., and Chang, P.R., Efficient Parallel Algorithm for Robot Inverse Dynamics Computation, paper presented at the Procedings of 1986 International Conference on Robotics and Automation, April, San Fransisco, CA.

Lee, C.S.G., and Ziegler, M., "Geometric Approach in Solving Inverse Kinematics of PUMA Robots," IEEE Transactions on Aerospace and Electrical Systems, v. AES-20, no. 6, November 1984.

Loranzo-Perez, T., "Robot Programming," Proceedings of IEEE, v. 71, no. 7, pp. 821-841, July 1983.

Ozaslan, K., The Near Minimum Time Control of a Robot Arm, Masters Thesis, Naval Postgraduate School, Monterey CA, December 1986.

Paul, R.P., Robot Manipulators. Mathmatics. Programming. and Control , MIT Press, 1981.

Thaler, G.J. and Stein, W.A., "Transfer Function and Parameter Evaluation for D-C Servomotors," Applications and Industry, January 1956.

BIBLIOGRAPHY

Ardayfio, D.D., and Pottinger, H.J., Computer Control of Robotic Manipulators, Mech. Eng., v. 104, no. 8 August 1982, pp. 40-45.

Asada, H., and Slotine, J.J., Robot Analysis and Control, John Wiley and Sons, 1986.

Craig, J.T., Introduction to Robotics: Mechanics and Control, Addison-Wesley, 1986.

Denavit, J., and Hartenberg, R.S., A Kinematic Notation For Lower Pair Mechanisms Based on Matrices, J. Appl. Mech. ASME, June 1955, pp. 215-221.

Derby, S. Simulating Motion Elements of General Purpose Robot Arms, International Journal of Robotics, v. 2, no. 1 pp. 3-12, 1983.

Ersu, E., and Nungesser, D., A Numerical Solution of the General Kinematic Problem, Technical Institute of Darnstadt, FRG.

Grossman, D.D., Programming a Computer Controlled Manipulator by Guiding Through The Motions, IBM J.T. Watson Res. Cen., Res. Rep RC6393, 1977 (declassified 1981).

Holm, R.E., Computer Path Control of an IR, Proc. of 8th ISIR, 1978, pp. 327-332.

Luh, J.Y.S., Walker, M.W., and Paul, R.P., On-Line Computational Scheme For Mechanical Manipulators, J. Dyn. Syst. Meas. Control, v. 102, 1980a, pp. 69-76.

Paul, R.P., Modelling, Trajectory Calculation and Servoing of a Computer Control Arm, Stanford University, AI Lab, Memo AIM 177, 1972.

Paul, R.P., Manipulator Cartesian Path Control, IEEE Trans. SYST. Man Cybernet., SMC-9, 1979, pp.702-711.

Suh, C.H. and Radcliffe, C.W., Kinematics and Machanisms Design, John Wiley and Sons, New York, 1978.

Takase, K., Paul, R.P., and Berg, E.E., A Structured Approach To Robot Programming and Teaching, paper presented at IEEE COMPSAC, Chicago, Il, November 1979.

Tasi, L., and Morgen, A.P., Solving the Kinematics of Most General 6 and 5 Degree of Freedom, paper presented at ASME Mechanism Conference, Boston, 7-10 October 1984, paper 84-DET-20.

Taylor, R.H., Planning and Execution of Straight Line Manipulator Trajectories, IBM J. Res. Dev., v. 23, no. 4, 1979, pp. 424-436.

Whitney, D.E., The Mathmatics and Coordinate Control of Prosthetic Arms and Manipulators, J. Dyn Syst. Meas. Control, December 1972, pp. 303-309.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Professor G.J. Thaler, Code 62Tr Department of Electrical Engineering Naval Postgraduate School Monterey, California 93943	5
5.	Professor H.A. Titus, Code 62Ti Department of Electrical Engineering Naval Postgraduate School Monterey, California 93943	1
6.	Lieutenant Steven G. Goodway c/o Bill Goodway 5310 SE Oetkin Road Milwaukie, Oregon 97267	5
7.	Professor D.L. Smith, Code 69Sm Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1

8.	Commander Naval Sea Systems Command Washington, D.C. 20361-0001	1
9.	Commander Naval Surface Weapons Center White Oak Silver Springs, Maryland 20910	1
10.	Director Naval Research Labotatory Washington, D.C. 20375	1
11.	Commander Naval Weapons Center China Lake, California 93555	1
12.	Chief of Naval Operations Attn: Code OP-03 Washington, D.C. 20350	1
13.	Chief of Naval Operations Attn: Code OP-04 Washington, D.C. 20350	1
14.	Dr. Russ Werneth Naval Surface Weapons Center G-40 Silver Springs, Maryland 20903-5000	1
15.	Mr. W. Butler Naval Sea Systems Command Attn: C90G Washington, D.C. 20362	1